Fast Online Changepoint Detection via Functional Pruning CUSUM statistics

Gaetano Romano, Idris Eckley, Paul Fearnhead, Guillem Rigaill

HAL Id: hal-04190669

https://hal.inrae.fr/hal-04190669v1

Submitted on 29 Aug 2023

Fast Online Changepoint Detection via Functional Pruning CUSUM Statistics

Gaetano Romano   G.ROMANO@LANCASTER.AC.UK
Department of Mathematics and Statistics,
Lancaster University,
Lancaster, United Kingdom, LA1 4YF

Idris A. Eckley   I.ECKLEY@LANCASTER.AC.UK
Department of Mathematics and Statistics,
Lancaster University,
Lancaster, United Kingdom, LA1 4YF

Paul Fearnhead   P.FEARNHEAD@LANCASTER.AC.UK
Department of Mathematics and Statistics,
Lancaster University,
Lancaster, United Kingdom, LA1 4YF

Guillem Rigaill   GUILLEM.RIGAILL@INRAE.FR
Université Paris-Saclay, CNRS, INRAE, Université d'Évry,
Institute of Plant Sciences Paris-Saclay (IPS2),
91405, Gif-sur-Yvette, France
and
Université d'Évry val d'Essonne, CNRS, INRAE,
Laboratoire de Mathématiques et Modélisation d'Evry,
91000, Evry, France

Abstract

Many modern applications of online changepoint detection require the ability to process high-frequency observations, sometimes with limited available computational resources. Online algorithms for detecting a change in mean often involve using a moving window, or specifying the expected size of change. Such choices affect which changes the algorithms have most power to detect. We introduce an algorithm, Functional Online CuSUM (FO-CuS), which is equivalent to running these earlier methods simultaneously for all sizes of windows, or all possible values for the size of change. Our theoretical results give tight bounds on the expected computational cost per iteration of FOCuS, with this being logarithmic in the number of observations. We show how FOCuS can be applied to a number of different changes in mean scenarios, and demonstrate its practical utility through its state-of-the-art performance at detecting anomalous behaviour in computer server data.

Keywords: Breakpoints; Changepoints; CUSUM; Online; Real-time analysis; FPOP.

1. Introduction

Over the previous decade, we have witnessed a renaissance of changepoint algorithms, and they can now be seen to make a difference in many real-world applications. Most of the

current literature focuses on an *a posteriori* analysis, having observed a series of data. Such an approach is often referred to as *offline* changepoint detection. However, as technology develops, the demand from several fields for *online* changepoint detection procedures has increased drastically over recent years. Examples include, but are not limited to, IT and cyber security (Jeske et al., 2018; Tartakovsky et al., 2012; Peng et al., 2004); detecting gamma-ray bursts in astronomy (Fridman, 2010; Fuschino et al., 2019); detecting earthquake tremors (Popescu and Aiordăchioaie, 2017; Xie et al., 2019); industrial processes monitoring (Pouliezos and Stavrakakis, 2013); detecting adverse health events (Clifford et al., 2015); and monitoring the structural integrity of aeroplanes (Alvarez-Montoya et al., 2020; Basseville et al., 2007).

The online setting raises computational challenges that are not present in offline changepoint detection. A procedure needs to be sequential, in the sense that one should process the observations as they become available, and at each iteration, one should make a decision whether to flag a changepoint based on the information to date. The procedure also needs to be able to run on a finite state machine for an indefinite amount of iterations, i.e. be constant in memory. Finally, a procedure should ideally be able to process observations, at least on average, as quickly as they arrive. Many online changepoint application settings have high-frequency observations, and some also have limited computational resources. For example, the ECG data in the 2015 PhysioNet challenge (Clifford et al., 2015) are sampled at 240Hz, while methods for detecting gamma-ray bursts (Fuschino et al., 2019) need to process high-frequency observations and be able to be run on small computers on board micro-satellites. A challenge with online changepoint algorithms is to meet such computational constraints whilst still having close to optimal statistical properties. This paper considers the univariate change in mean problem within precisely this setting.

Current online changepoint methods with a linear computational cost include the method of Page (1955) that assumes knowledge both of the pre-change and post-change mean; or moving window methods such as MOSUM. Assuming the pre-change mean is known is reasonable in many applications as there will be substantial data to estimate this mean (though see discussion in Gösmann et al., 2019). However, the method of Page (1955) can lose power if the assumed size of the change is wrong. For example, this method can have almost no power to detect changes that are less than half the size of the assumed change. Similarly, moving window methods can perform poorly if the window size is inappropriate for the size of the change. For example, a small window size will result in little power at detecting small changes, whilst too large a window will lead to delays in detecting larger changes. See Section 2.1 for an example of these issues.

An alternative approach, with more robust statistical properties, is to, e.g., apply a moving window but consider all possible window sizes. In the known pre-change mean setting this is known as the Page-cusum approach (Kirch and Weber, 2018) and is the approach of Yu et al. (2020) for the case of an unknown pre-change mean. The theoretical results in Yu et al. (2020) demonstrate the excellent statistical properties of such a method. However current exact implementations of this idea have a computational cost per iteration that is linear in the number of observations and thus have an overall quadratic computational cost. The problem of deriving an efficient implementation capable of solving the Page-CUSUM statistic dates back to the past century. Basseville and Nikiforov (1993) state that "[...] the GLR [Generalized Likelihood Ratio - Page's method] algorithm is computationally com-

plex", before presenting some approximations. More recently, Yu et al. (2020) commented on the challenge of developing faster algorithms with good statistical guarantees. We introduce an algorithm to calculate the GLR exactly in an efficient way, which we call Functional Online CuSum (FOCuS). We develop FOCuS for detecting changes in mean in univariate data under a Gaussian model, and it can be applied to settings where either the pre-change mean is known or unknown. FOCuS recursively updates a piecewise quadratic, whose maximum is the test statistic for a change. We show that the amortized cost for solving the recursion is constant per iteration. Maximising the function has a computational cost proportional to the number of components in the piecewise quadratic, and the average number of components increases with the logarithm of the number of observations. We can obtain algorithms with a constant computation per iteration by, when necessary, restricting the maximisation to a fixed number of components. We develop one such approximate version of FOCuS and show empirically that it has almost identical statistical performance to an exact implementation. In the unknown pre-change mean case FOCuS implements the method of Yu et al. (2020), and our exact implementation of FOCuS can analyse 1 million observations in less than a second on a common personal computer.

Much research on online changepoint methods has looked at how to implement methods so that they have well-characterised performance under the null hypothesis of no change. There are two distinct criteria for quantifying a method's behaviour under the null, one is the average run length (Reynolds, 1975) which is the expected number of observations until we detect a change. The other is a significance level – the probability of ever detecting a change if the method is run on infinitely long data. In practice, these two criteria affect the choice of threshold for a detection method. If we wish to control the significance level then we need a threshold that increases with the number of observations (see e.g. Kirch and Kamgaing, 2015), whereas if we wish to control the average run length we can use a fixed threshold. The FOCuS algorithm can be used with either approach – but for simplicity we will only use a fixed threshold in this paper.

The outline of the paper is as follows. In Section 2 we consider the challenge of detecting a univariate change in mean when the pre-change mean is assumed known. We present the FOCuS algorithm which can be viewed as implementing the procedure of Page (1955) simultaneously for all possible sizes of change. Our main theoretical result shows that FOCuS achieves this with an average computational cost per iteration that is logarithmic in the number of data points. Our bound on the average per iteration cost is tight, and processing the one-millionth observation roughly equates to the cost of evaluating 15 quadratics. In Section 3 we study FOCuS for the case where the pre-change mean is unknown. This algorithm can be viewed as implementing the statistical tests of Yu et al. (2020). Interestingly, whilst their algorithms are either exact but with a linear cost per iteration, or approximate with a cost that is logarithmic in the number of iterations, the FOCuS algorithm is both exact and has an average cost that is logarithmic per iteration. We do not present any statistical theory for FOCuS, as this is covered in Yu et al. (2020). In Section 4 we then introduce an extension of the FOCuS algorithm to the scenario of detecting changes in the presence of outliers. In Section 5 we show a monitoring application for FOCuS on some AWS Cloudwatch server instances. Finally, the paper concludes with a discussion, which includes evaluating the use of running FOCuS independently on multiple data streams and then combining the results to detect a change that may jointly affect multiple streams.

Software implementing FOCuS and the code for our simulation study is available at
`https://github.com/gtromano/FOCuS`.

2. Known pre-change mean

2.1 Problem set-up and background

Consider the problem of detecting a change in mean in univariate data. We will let x_t denote the data at time t, for $t = 1, 2, \ldots$. We are interested in online detection, that is after observing each new data point we wish to decide whether or not to flag that a change has occurred. We first assume that the pre-change mean is known. Often the methods below are implemented in practice using a plug-in estimator for the pre-change mean that is calculated from training data.

Whilst there are many different approaches to online detection of change (see Veeravalli and Banerjee, 2014, for some examples), a common approach (see Kirch and Weber, 2018) is to use a cumulative sum of score statistics, also known as a CUSUM based procedure. Assume we model our data as coming from a parametric model with density $f(x; \mu)$ and denote the pre-change mean as μ_0. Define the score statistic of an observation x as

$$H(x, \mu) = \frac{\partial \log f(x; \mu)}{\partial \mu}.$$

Then if there is no change prior to time n

$$\mathrm{E}\left(H(X_i, \mu_0)\right) = 0, \text{ for } i = 1, \ldots, n.$$

Thus evidence of a change prior to time n can be obtained by monitoring the absolute values of partial sums of these score statistics, which we denote as

$$S(s, n) = \sum_{i=s+1}^{n} H(x_i, \mu_0).$$

The idea is that these partial sums should be close to 0 if there is no change, and conversely diverge from zero if there is a change.

For ease of presentation, and to make ideas concrete, in the following we will consider the case where we have a Gaussian model with unit variance for the data. In this case $H(x, \mu) = (x - \mu)$. Also as we are assuming μ_0 is known, then without loss of generality we can set $\mu_0 = 0$.

There have been a number of different choices of partial sums that we can monitor. For detecting a change after observing x_n, Kirch and Weber (2018) highlight the following statistics:

$$\text{CUSUM} \qquad C(n) = \frac{1}{\sqrt{n}} |S(0, n)|; \tag{1}$$

$$\text{MOSUM} \qquad M_w(n) = \frac{1}{\sqrt{w}} |S(n - w, n)|; \tag{2}$$

$$\text{mMOSUM} \qquad M_k^{(\mathrm{m})}(n) = \frac{1}{\sqrt{nk}} |S(n - \lfloor kn \rfloor, n)|; \tag{3}$$

$$\text{Page-CUSUM} \qquad P(n) = \max_{0 \le w < n} \frac{1}{\sqrt{w}} |S(n - w, n)|. \tag{4}$$

The scale factor in each case is to normalise the cumulative sum $S(\cdot, \cdot)$, with the aim of standardizing its variance. In each case, we would compare the statistic at time n with some appropriate threshold, and detect a change prior to n if the statistic is above the threshold. As discussed in the introduction the choice of threshold impacts the properties of the test under the null. It is possible to choose thresholds that are constant or that increase as n increases (Kirch and Weber, 2018), but for simplicity we will use constant thresholds throughout.

The standard CUSUM statistic uses the partial sum of score statistics to time n. For both the MOSUM procedure (Eiauer and Hackl, 1978; Chu et al., 1995) and mMOSUM procedure (originating in Chen and Tian, 2010) we need to specify a tuning parameter. The MOSUM method uses the partial sum over a window of the most recent $w > 0$ observations, whilst the mMOSUM fixes some proportion $0 < k < 1$ and uses the partial sum over the most recent proportion k of the observations. All three of these statistics are online, in that there is only an $O(1)$ update of the statistics as we process each new data point. The Page-CUSUM maximises over all possible partial sums ending at time n.

To illustrate the difference between CUSUM, MOSUM and Page-CUSUM we implement these methods to detect a change at time t to some mean μ_1 for different values of t and μ_1. We calculate the different test statistics under our model where the data is independent Gaussian with unit variance. In Figure 1 we compare the detection delay of each statistic. In Figure 1a, we simulated data with a change after 1000 observations. The size of the change is chosen to give high power for the window size of the MOSUM procedure. MOSUM and Page-CUSUM tend to detect a change quickly. In the second example, shown in Figure 1b, we reduce the magnitude of the change. Here the MOSUM test loses power substantially and fails to detect a change. Whilst increasing the window would detect the change, using a larger window size would increase the detection delay in the first example. In our final example, Figure 1c, we have the same size of change as the first example, however, the change now occurs after 8,000 observations. In this case, we see that the CUSUM statistic behaves poorly. This is because the CUSUM statistic has to average the signal from data after the change with all the data prior to the change, and this reduces the power of the test statistic. This is particularly the case when there is substantial data prior to the change. Both MOSUM and Page-CUSUM perform as in the first example. Whilst we do not show the performance of mMOSUM in these examples, it shares a similar sensitivity to the choice of window proportion, k, as the MOSUM does for window size.

The Page-CUSUM approach tries to avoid the issues with choosing a window size within the MOSUM method, and is equivalent to maximising the MOSUM statistic over w. However current implementations of Page-CUSUM are not online. Indeed, the computational cost of calculating $\max_{0 \leq s < n} |S(s, n)|$ increases linearly with n, resulting in an $O(n^2)$ computational complexity.

An alternative approach (Page, 1954, 1955) to detecting the change is based on sequentially applying a likelihood ratio test under an assumed value for the post-change mean, μ_1. Under our Gaussian model, with the pre-change mean being 0 and variance 1, the contribution to the log-likelihood ratio statistic from a single data point, x_t, is twice the difference in log-likelihood if the data has mean μ_1 as opposed to mean 0:

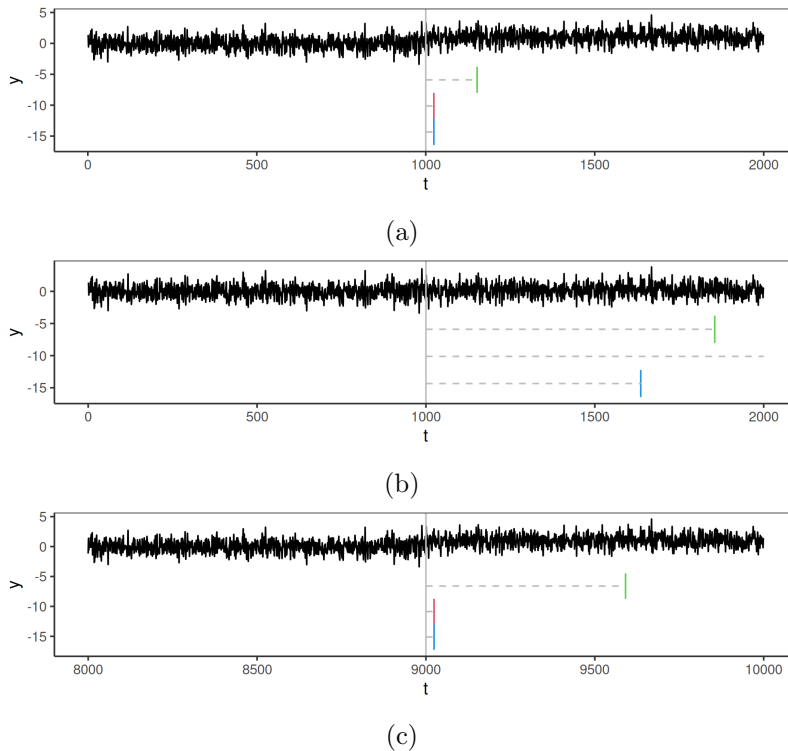$$LR(x_t, \mu_1) = 2\mu_1 \left(x_t - \frac{\mu_1}{2} \right)$$

(a)

(b)

(c)

Figure 1: Detection delays of CUSUM (in green), MOSUM (in red, with $w = 50$) and Page-CUSUM (in blue) on three sequences. The sequences were generated in the following way: (a) a sequence of 2000 observations with a change of size 1 at 1000; (b) similar to (a) but with a change in the mean of 0.2; (c) similar (a) again, but with an additional 8×10^3 observations at the start of the sequence. Thresholds were tuned accordingly to the simulation study in Section 2.3. The solid grey line refers to the true changepoint location, the dashed segments to the detection delays of the procedures.

As is common in this setting, we will work with a test statistic that is half the log likelihood-ratio statistic (though obviously using such a test statistic is equivalent to using the log likelihood-ratio test statistic). At time n, our test statistic for a change at time s is thus half the sum of the $LR(x_t, \mu_1)$ terms from $t = s + 1, \ldots, n$. As we do not know the time of the change, we maximise over s:

$$\mathcal{Q}_{n,\mu_1} = \max_{0 \leq s \leq n} \sum_{t=s+1}^{n} \mu_1 \left(x_t - \frac{\mu_1}{2} \right),$$

where we use the convention that the sum from $s = n+1$ to n is 0. We will call this statistic the *sequential-Page statistic*.

Whilst $\mathcal{Q}_{n,\mu_1}$ involves a sum over n terms, Page (1954) showed that we can calculate $\mathcal{Q}_{n,\mu_1}$ recursively in constant time as:

$$\mathcal{Q}_{n,\mu_1} = \max \left\{ 0, \ \mathcal{Q}_{n-1,\mu_1} + \mu_1 \left(x_n - \frac{\mu_1}{2} \right) \right\}. \tag{5}$$

6

One issue with the sequential-Page statistic is the need to specify μ_1, and a poor choice of μ_1 can substantially reduce the power to detect a change. This is similar to the choice of window size for MOSUM. To partially overcome this, in both cases we can implement the methods multiple times, for a grid of either window sizes or values of μ_1. Obviously, this comes with an increased computational cost.

Lorden (1971) suggests a procedure for extending Page's method to test for all changes bigger than a minimum size of change. To ease exposition assume that we wish to test for a positive change, with a similar approach being applicable for a negative change. Let μ_1 be the minimum size of change. Lorden (1971) uses a statistic that is the maximum of the sequential-Page statistics over all changes larger than μ_1. This procedure is based on running Page's method for μ_1, and recording the reset times, that is the times at which the sequential-Page statistic is 0. Lorden (1971) then uses the property that at any time t, the maximum of the sequential-Page statistic will be the maximum of half the log likelihood-ratio statistic for a change at time s, where we maximise over all s greater than or equal to the most recent reset time. The computational cost of this at time t is proportional to the number of time points since the last reset.

This idea has some similarity to the FOCuS^0 algorithm we introduce next, in that it tries to cleverly choose which tests to perform. But FOCuS^0 does not require a minimum size of change to be specified; and FOCuS^0 is more efficient at deciding which tests to calculate, so that it is computationally more efficient (unless the minimum size of change of Lorden's algorithm is around 0.5 or higher, see Appendix C.1).

2.2 FOCuS^0: solving the Page recursion for all μ_1

Our idea is to solve the sequential-Page recursion simultaneously for all values of the post-change mean. To this end, we re-write (5) in terms of a recursion for a function $Q_n(\mu)$ of the post-change mean μ. We then have $Q_0(\mu) = 0$ and for $n = 1, \dots,$

$$Q_n(\mu) = \max\left\{0,\ Q_{n-1}(\mu) + \mu\left(x_n - \frac{\mu}{2}\right)\right\}. \tag{6}$$

We would then use $\max_\mu Q_n(\mu)$ as our test statistic. It is straightforward to see that for any μ_1, $Q_n(\mu_1) = \mathcal{Q}_{n,\mu_1}$. Thus if we can efficiently calculate the function $Q_n(\mu)$ then our test statistic is equivalent to the maximum value of the sequential-Page statistic over all possible choices of the post-change mean. Furthermore, the following proposition (see, for instance, Example 2.4.3 in Basseville and Nikiforov, 1993) shows that this test statistic is equivalent to the Page-CUSUM statistic (4), or equivalently the maximum of the MOSUM statistic (2) over all possible windows.

Proposition 1 *The maximum of $Q_n(\mu)$ satisfies*

$$\max_\mu Q_n(\mu) = \frac{1}{2}P(n)^2 = \frac{1}{2}\max_w M_w(n)^2,$$

where $P(n)$ is the Page-CUSUM statistic and $M_w(n)$ is the MOSUM statistic with window size w.

The proof of this can be found in Appendix A.

Algorithm 1: FOCuS^0 (one iteration)

    Data: x_n the data at time n; $Q_{n-1}(\mu)$ the cost function from the previous
        iteration.

    Input: $\lambda > 0$

1 $Q_n(\mu) \leftarrow \max\left\{0,\ Q_{n-1}(\mu) + \mu\left(x_n - \frac{\mu}{2}\right)\right\}$;   `// Algorithm 2 :  amortized O(1)`

2 $\mathcal{Q}_n \leftarrow \max_\mu Q_n(\mu)$;   `// Theorem 4 :  average O(log(n))`

3 **if** $\mathcal{Q}_n \geq \lambda$ **then**

4 | **return** n *as a stopping point*;

5 **end**

6 **return** $Q_n(\mu)$ *for the next iteration.*

A description of the resulting algorithm for online changepoint detection is given in Algorithm 1. We call this the Functional Online CuSUM (FOCuS) algorithm. To be able to distinguish this version, which assumes a known pre-change mean, from the version we introduce in the next section, we call Algorithm 1 FOCuS^0. The FOCuS^0 algorithm is only useful if it is computationally efficient, and in particular if we can implement Steps 1 and 2 efficiently. These steps correspond to solving the recursion in (6) to obtain the function $Q_n(\mu)$ from $Q_{n-1}(\mu)$ and then maximising the function $Q_n(\mu)$. Below, we describe each of these steps in turn, and present results on their average computational cost.

2.2.1 STEP 1: UPDATING THE INTERVALS AND QUADRATICS

For Step 1 of Algorithm 1 we propose to update the function $Q_n(\mu)$ separately for $\mu > 0$ and $\mu < 0$. These can be updated in an identical manner, so we will only describe the update for $\mu > 0$. We will use the fact that (6) maps piecewise quadratics to piecewise quadratics, and hence $Q_n(\mu)$ will be piecewise quadratic (see Maidstone et al., 2017, for a similar idea) and can be stored as a list of ordered intervals of μ together with the coefficients of the quadratic for $Q_n(\mu)$ on that interval. Let $S_t = \sum_{j=1}^{t} x_j$ be the sum of the first t data points. At time n the quadratic introduced at iteration τ will be of the form

$$\mu\left(\sum_{t=\tau+1}^{n} x_t - (n-\tau)\frac{\mu}{2}\right) = \mu\left((S_n - S_\tau) - (n-\tau)\frac{\mu}{2}\right). \tag{7}$$

Thus, if at time n we know n and S_n, its coefficients can be calculated if we store τ and S_τ. This information stored for the quadratic does not need to be updated at each iteration.

So at any time t we are able to produce a compact summary of the function $Q_t(\mu)$ by storing a set of triples (τ_i, s_i, l_i) for each of the k_t quadratics that define the piecewise quadratic $Q_t(\mu)$. The entries are τ_i, the time at which the ith quadratic is introduced, $s_i = S_{\tau_i}$ the sum of observations up to τ_i, and l_i the left-hand point of the interval of μ for which the ith quadratic is optimal. The quadratics are ordered so that $0 = l_1 < \cdots < l_{k_t}$, thus for $i < k_t$ the ith quadratic is optimal for $\mu \in [l_i, l_{i+1})$, with the k_tth quadratic optimal for $\mu \in [l_{k_t}, \infty)$. Furthermore, if $l_i < l_j$ then $\tau_i < \tau_j$, as quadratics introduced more recently will be optimal for larger values of μ than quadratics introduced less recently. (This can be shown as all quadratics go through the origin, and the μ^2 coefficient of the ith quadratic will be larger than that of the jth quadratic.)

Now, consider an iteration to calculate the function $Q_n(\mu)$ from the function $Q_{n-1}(\mu)$. This is given in Algorithm 2. We will have currently stored the k_{n-1} triples associated with the quadratics that define $Q_{n-1}(\mu)$. We split the recursion into two. First we calculate the intermediate function $Q_n^*(\mu) = Q_{n-1}(\mu) + \mu(x_n - \mu/2)$. This will update the coefficients of each of the k_{n-1} quadratics that define $Q_{n-1}(\mu)$. However as we are defining the quadratics in terms of the summary statistics of the sum of the data points (7), this is achieved by updating the sum of all data points $S_n = S_{n-1} + x_n$.

Second we calculate $\max\{0, Q_n^*(\mu)\}$. To do this we first add a quadratic corresponding to the zero-line. This will have triple (n, S_n, l), for some l such that $Q_n^*(l) = 0$, that we need to calculate.

A key observation is that the difference between a quadratic introduced at iteration τ and the zero-line gives that the zero-line is better on the interval

$$\left[2 \sum_{t=\tau+1}^{n} \frac{x_t}{(n-\tau)} \ , \ +\infty \right) = \left[2 \frac{S_n - S_\tau}{(n-\tau)} \ , \ +\infty \right). \tag{8}$$

Considering all τ we get that the zero-line is better than all others on

$$\left[2 \max_\tau \frac{S_n - S_\tau}{(n-\tau)} \ , \ +\infty \right).$$

Therefore we have that the final component for the triple defining the zero-line is

$$l = 2 \max_\tau \frac{S_n - S_\tau}{(n-\tau)}. \tag{9}$$

If we can calculate l then it immediately follows that any quadratic with $l_k > l$ can be removed. And any quadratic with $l_k < l$ will be unaffected (the interval on which they are optimal may be changed, but the triple that needs to be stored for the quadratic will be unaffected). By using the ordering of the quadratics mentioned above, we find l by comparing the zero-line with each of each quadratic in turn, starting with k_{n-1}th quadratic and stepping through them in decreasing order. If we are considering the ith quadratic we check whether $Q_n^*(l_i) < 0$ or not. This is equivalent to checking whether the ith quadratic is less than 0 at l_i, or not. If it is we remove the quadratic and move to the $(i-1)$th quadratic (or stop if $i = 1$). If $Q_n^*(l_i) > 0$ then $l > l_i$ and we find l as the positive value of μ such that the ith quadratic is equal to 0.

A pictorial representation of the algorithm is shown in Figure 2. The condition for l is related to the slope of the random walk S_t, $t = 1, \ldots, n$. As shown for an example in Figure 3, the quadratics that we keep are related to a subset of the vertices of the lower convex hull of this random walk (this property is proven formally in the proof of Theorem 4), a property that will be useful in bounding the computational complexity of the algorithm. Furthermore, Algorithm 2 is closely related to Melkman's algorithm for finding the convex hull of a set of points (Melkman, 1987).

We can show that Algorithm 2 has an amortized per-iteration cost that is $O(1)$. The intuition is that each quadratic is added once and removed once, and otherwise unchanged. Thus the average per-iteration cost is essentially the cost of adding and removing a quadratic.
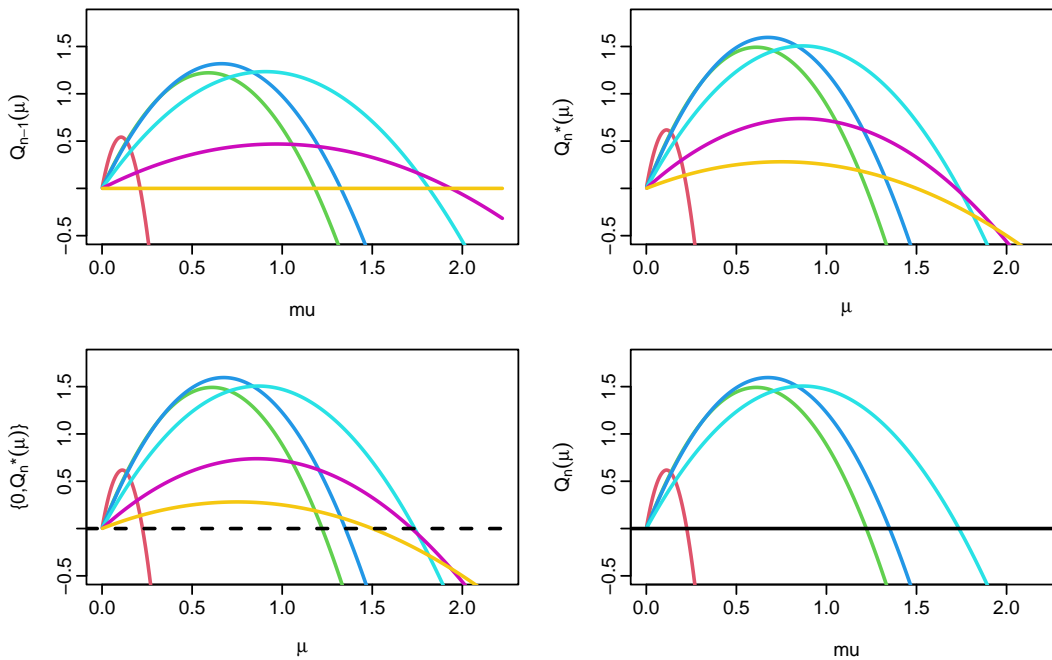
9

Figure 2: Example of one iteration of FOCuS^0. (Top left) The output at time $n-1$ is the function $Q_{n-1}(\mu)$ which is the maximum of a set of quadratics. In this example it is the maximum of 6 quadratics, one of which is the zero-line. In the algorithm, each quadratic is represented by a triple (τ, s, l), whose elements are the time the quadratic is introduced, the sum of observations at that time, and the smallest μ value for which it is optimal. A key property is that these are ordered so the quadratics introduced early have a higher coefficient of μ^2 and are thus optimal for lower values of μ. (Top right) The function $Q_n^*(\mu) = Q_{n-1}(\mu) + \mu(x_n - \mu/2)$. This is still the maximum of six quadratics, but the coefficients have changed from the top left plot. In practice, this involves no need to update the triples for the quadratics. (Bottom left) We introduce the zero-line (black dashed line), which is the quadratic introduced at time n. We compare the zero-line with each current quadratic by seeing which is larger at the lowest μ value for which the quadratic was optimal, starting with the most recent quadratics (i.e. yellow, then purple etc.). If the quadratic is below the zero line it is no longer optimal and can be removed. For the first quadratic that is above the zero-line, we calculate the non-zero value of μ where the quadratic and zero-line intercept. The maximum of this and 0 is the smallest μ value for which the zero-line is optimal. (Bottom right) The function $Q_n(\mu)$ after removing the two quadratics that are no longer optimal.
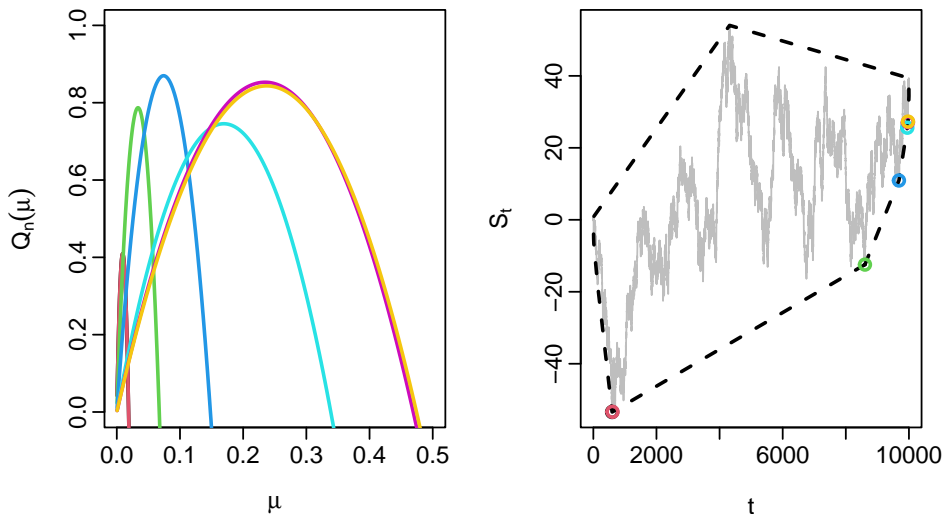
Figure 3: Example showing that the quadratics kept by FOCuS^0 are introduced at times related to vertices of the convex hull of the random walk $S_t = \sum_{i=1}^{t} x_i$. (Left) Plot of quadratics defining $Q_n(\mu)$. (Right) Plot of S_t as a function of t (grey), and the convex hull of the points (t, S_t) (dashed line). We have circled all points (t, S_t) associated with the time each quadratic in the left-hand plot was introduced (with the same colouring), and these correspond to vertices of the convex hull. All vertices lie on the lower convex minorant because we are considering a positive change in mean. We only keep quadratics associated with vertices to the left of sides of the convex hull with positive gradient due to the condition that the post-change mean is larger than the pre-change mean of 0. Changing the pre-change mean thus affects only which of the vertices on the lower minorant correspond to quadratics that are kept by FOCuS^0

Algorithm 2: Algorithm for $\max\{0, \; Q_{n-1}(\mu) + \mu(x_n - \mu/2)\}$ for $\mu > 0$

Data: $Q_n^+(\mu) = Q$ an ordered set of triples $\{q_i = (\tau_i, \; s_i, \; l_i) \; \forall \; i = 1, \ldots, \; k\}$,
x_n and S_{n-1}

1 $S_n \leftarrow S_{n-1} + x_n$;   // update cumulative sum

2 $q_{k+1} \leftarrow (\tau_{k+1} = n, \; s_{k+1} = S_n, \; l_{k+1} = \infty)$;   // new quadratic

3 $i \leftarrow k$;

4 while $2(s_{k+1} - s_i) - (\tau_{k+1} - \tau_i)l_i \leq 0 \;\;$ *and* $i \geq 1$ **do**

5 | $\;\; i \leftarrow i - 1$;

6 end

7 $l_{k+1} \leftarrow \max\{0, \; 2(s_{k+1} - s_i)/(\tau_{k+1} - \tau_i)\}$;   // update new border

8 if $i \neq k$ **then**

9 | $\;\; Q \leftarrow Q \setminus \{q_{i+1}, \; \ldots, q_k\}$;   // pruning old quadratic

10 end

11 return $\{Q, q_{k+1}\}, \;\;\; S_n$

Theorem 2 *The worst case complexity of Algorithm 2 for any data* $x_1, \ldots, x_T$ *is* $O(T)$ *and its amortized complexity per iteration is* $O(1)$.

Proof: At iteration n, let k_n be the number of quadratics input, and let c_n be the number of times the while statement is evaluated. Let C_1 be the cost of steps 1 to 3 and 7 to 11, and C_2 be the cost of one set of one evaluation of steps 4 to 6. Then the computational cost of one iteration of Algorithm 2 is $C_1 + c_n \times C_2$.

The key observation is that $k_{n+1} = k_n - (c_n - 1) + 1$. That is if we repeat Steps 4 to 6 c_n times then we will remove $c_n - 1$ quadratic in Step 9 and add one quadratic in Step 11. Furthermore $k_1 = 0$ and k_{T+1} is the number of quadratics for $Q_T(\mu)$. Thus the total computational cost is

$$\sum_{n=1}^{T}(C_1 + c_n C_2) = C_1 T + C_2 \sum_{n=1}^{T} c_n = C_1 T + C_2 \sum_{n=1}^{T}(2 + k_n - k_{n+1}).$$

Due to the cancellations in the telescoping sum and the fact that $k_1 = 0$ we have

$$\sum_{n=1}^{T} c_n = 2T - k_{T+1} \leq 2T$$

Thus the theorem holds $\square$

As the overall computational cost is linear in T, the expected cost per iteration must be constant. The proof gives a form for the overhead in terms of the operations in Algorithm 2. In practice, this cost is observed to be negligible relative to the cost of step 2 of Algorithm 1, namely that of maximising $Q_n(\mu)$.

2.2.2 Step 2: Maximisation

To implement Step 2 of Algorithm 1 we first use the trivial observation that if $x_n > 0$ then $Q_n(\mu) < Q_{n-1}(\mu)$ for all $\mu < 0$. Thus to check if $\max_\mu Q_n(\mu) \geq \lambda$ we need only check this

for $\mu > 0$. Similarly if $x_n < 0$ then we need only check for $\mu < 0$. To perform the check we just loop over all quadratics stored for either $\mu > 0$ or $\mu < 0$, and for each one check if its maximum is greater than λ. For a quadratic with stored triplet (τ, s, l) this involves checking whether

$$(S_n - s)^2 \geq 2\lambda(n - \tau). \tag{10}$$

If we flag a change at time n, then we can also output the value of τ corresponding to the quadratic whose maximum is largest, and this will be an estimate of the time of the change. The computational cost is thus proportional to the number of quadratics that are stored and can be bounded using the following result.

Theorem 3 *Let $x_1, \ldots, x_T, \ldots$ be a realization of the process $X_i = \mu_i + \epsilon_i$ where ϵ_i are independent, identically distributed continuous random variables with mean 0. Let the number of quadratics stored by FOCuS^0 for $\mu > 0$ at iteration T be $\#\mathcal{I}_{1:T}^0$. Then if μ_i is constant*

$$E(\#\mathcal{I}_{1:T}^0) \leq (\log(T) + 1),$$

while if μ_i has one change prior to T then

$$E(\#\mathcal{I}_{1:T}^0) \leq 2(\log(T/2) + 1).$$

The proof of this can be found in Appendix B. The key idea is to use a one-to-one correspondence between each quadratic we need to keep and the vertices of the convex hull of the random walk $S_t = \sum_{i=1}^{t} x_i$ that is shown in Figure 3. Standard results (Andersen, 1955) give bounds on the vertices of a convex hull of a random walk where the increments are exchangeable.

By symmetry, the same result holds for the number of quadratics stored for $\mu < 0$. The conditions of the data generating mechanism are weak – as the distribution of the noise can be any continuous distribution providing the noise is independent. The theorem shows that the expected per-iteration time and memory complexity of FOCuS^0 at time T is $O(\log T)$. Furthermore, the expected per-iteration cost is essentially equal to checking (10) $\log(n) + 1$ times if there has not been a change, and $2(\log(n/2) + 1)$ times if there has been an, as yet, undetected change. A change of fixed size is detected in $O(1)$ iterations, and thus the overall computational time, for large T, will be dominated by the cost of iterations prior to the changepoint. For data of size one million, the bound on the number of quadratics is less than 15. For FOCuS^0, this bound can be improved upon. The bound is based on all quadratics that are stored corresponding to vertices on the convex minorant of the random walk $S_t = \sum_{i=1}^{t} x_i$, but as shown in Figure 3, not all vertices on the convex minorant correspond to quadratics that are stored. In fact, a simple symmetry argument suggests that only half of them do. This is validated empirically in Appendix C.2.

The FOCuS^0 algorithm is not strictly online, due to the cost per iteration not being bounded. But it is simple to introduce an approximation that is online. Assume we have a constraint that means we can find the maximum of at most P quadratics per iteration. A simple approximation is to introduce a grid of points $\pm m_p$ for $m_p \in \mathbb{R}^+$, $p = 1, \ldots P$. There are then two natural approaches. One is that if we have $P + 1$ quadratics stored we prune to P quadratics by removing the first quadratic whose interval does not contain a grid point. Alternatively, we can keep all quadratics but only find the maximum of the

quadratics whose interval contains a grid point. The advantage of this latter approach is that it avoids any approximations to the function $Q_n(\mu)$ which could propagate to future values of the functions $Q_t(\mu)$ for $t > n$. Both these methods would dominate using the sequential-Page approach that used the same grid for μ_1 values. For example, if $\tilde{Q}_n(\mu)$ denotes the approximation to $Q_n(\mu)$ using the first approach, then we have $\tilde{Q}_n(\mu_1) = \mathcal{Q}_{n,\mu_1}$ for all μ_1 in our grid. Thus, for either method, the maximum value of the quadratics evaluated by FOCuS^0 will be equal to or greater than the maximum of the sequential-Page statistics at the grid points. As with sequential-Page, it is sensible to use a geometric scaling for the grid points $\pm m_p$ (see below).

2.3 Simulation Study

We study the average run length and the detection delay of the FOCuS^0 procedure and its approximation (introduced at the end of Section 2.2), the sequential Page-CUSUM statistics from equation (5), and the MOSUM procedure from equation (2) in case of a pre-change mean known at 0.

The study is structured as follows: For the Page recursion we employ a geometric grid (as recommended by Chen et al., 2022). We use a 10 point grid as that is equivalent to the expected number of intervals stored in FOCuS^0 over a sequence of one hundred thousand observations. To see the potential benefits of using a finer grid, we also use a 20 point grid. We call these two approaches Page-20p, and Page-10p. We evaluate MOSUM over a set of 20 window sizes, with these sizes geometrically increasing, and chosen to give MOSUM greatest power for the size of changes specified by the grid used for Page-20p. We also measure performance of the FOCuS^0 approximation on the 10 points grid used for Page-10p. We call this approximation FOCuS^0-10p.

For each of these methods, we first estimate the run length as a function of the threshold based on data, with no change, of length two million observations. We average the results across 100 different replicates and summarise them in Figure 4. For each method we choose the threshold that gives an average run length of 1×10^6 observations, and evaluate the detection delay with this threshold over a range of different change magnitudes. To generate profiles with changes we superimpose on the previous 100 null profiles a piece-wise constant signal with a change at 1×10^5. For simplicity we only show changes with a positive increase in magnitude, however the study extends to negative changes too.

In Figure 5 we report log-ratios of the average detection delay for pairs of methods against the different change magnitudes. The most striking comparison is between FOCuS^0 and Page-10p. The relative performance of the methods depends on how the size of change matches with grid used for Page-10p. If these match exactly, then Page-10p has a slightly smaller average detection delay because it has a smaller threshold (see Figure 4). However as we move to changes that are different from the grid points, the Page-10p method loses power relative to FOCuS^0 and the latter can be substantially faster at detecting a change. Once we increase the grid size to 20, we have similar qualitative patterns but now the quantitative differences in performance are small, as is the difference between FOCuS^0 and FOCuS^0-10p. In general, the biggest gains of FOCuS^0 over the grid-based methods are seen for changes that are smaller than the minimum jump size, or larger than the maximum jump
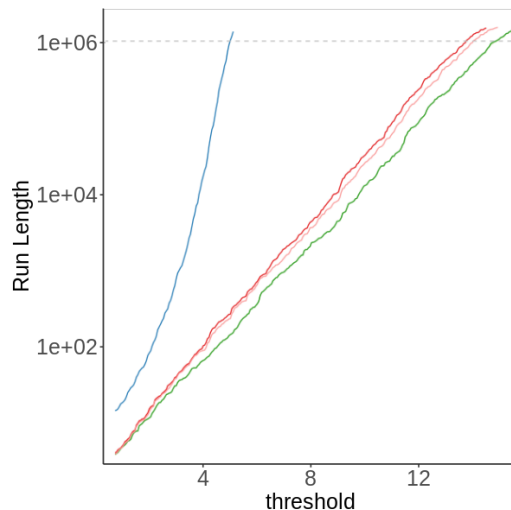
Figure 4: Average Run Length against threshold for FOCuS^0 and FOCuS^0-10p (both identical, and shown in green), Page-20p (pink), Page-10p (red) and MOSUM (blue). Results are averaged across 100 simulations. We use a log scale on the y axis.

size of the grid. Finally, we see that MOSUM gives noticeably the worst performance of all methods.

3. Unknown pre-change mean

Assume we are observing a sequence of observations $x_1, \ldots, x_n$ distributed as a $N(\mu_0, \sigma)$ prior to any change, and as $N(\mu_1, \sigma)$ after the change, with σ known and $\mu_1 \neq \mu_0$. As before, without loss of generality, we will assume $\sigma = 1$. As suggested by Yu et al. (2020), we can base a test for a change on the likelihood ratio statistic.

$$LR_n = \max_{\substack{\tau \in \{1,\ldots,n-1\} \\ \mu_0, \mu_1 \in \mathbb{R}}} \left\{ -\sum_{t=1}^{\tau}(x_t - \mu_0)^2 - \sum_{t=\tau+1}^{n}(x_t - \mu_1)^2 \right\} - \max_{\mu \in \mathbb{R}} \left\{ -\sum_{t=1}^{n}(x_t - \mu)^2 \right\}. \quad (11)$$

Yu et al. (2020) present finite-sample results that demonstrate the statistical optimality of such a test. They also present algorithms for evaluating this test statistic. Their fastest algorithm that avoids any approximation is $O(n)$ in computational complexity per iteration while being $O(n)$ in storage, which makes their methodology infeasible to a true online setting. For the rest of this paper, we will refer to that algorithm as Yu-CUSUM.

To be consistent with, and show the links to, how we calculated the test statistic in the pre-change mean known case, we introduce the following functions,

$$\mathcal{Q}_{\tau,n}(\mu_0, \mu_1) = -\frac{1}{2}\sum_{t=1}^{\tau}(x_t - \mu_0)^2 - \frac{1}{2}\sum_{t=\tau+1}^{n}(x_t - \mu_1)^2.$$

This is the log-likelihood for a model with a change at τ from mean μ_0 to μ_1. If $\mu_1 = \mu_0$ this reduces to the log-likelihood for the data having mean μ_0 throughout.
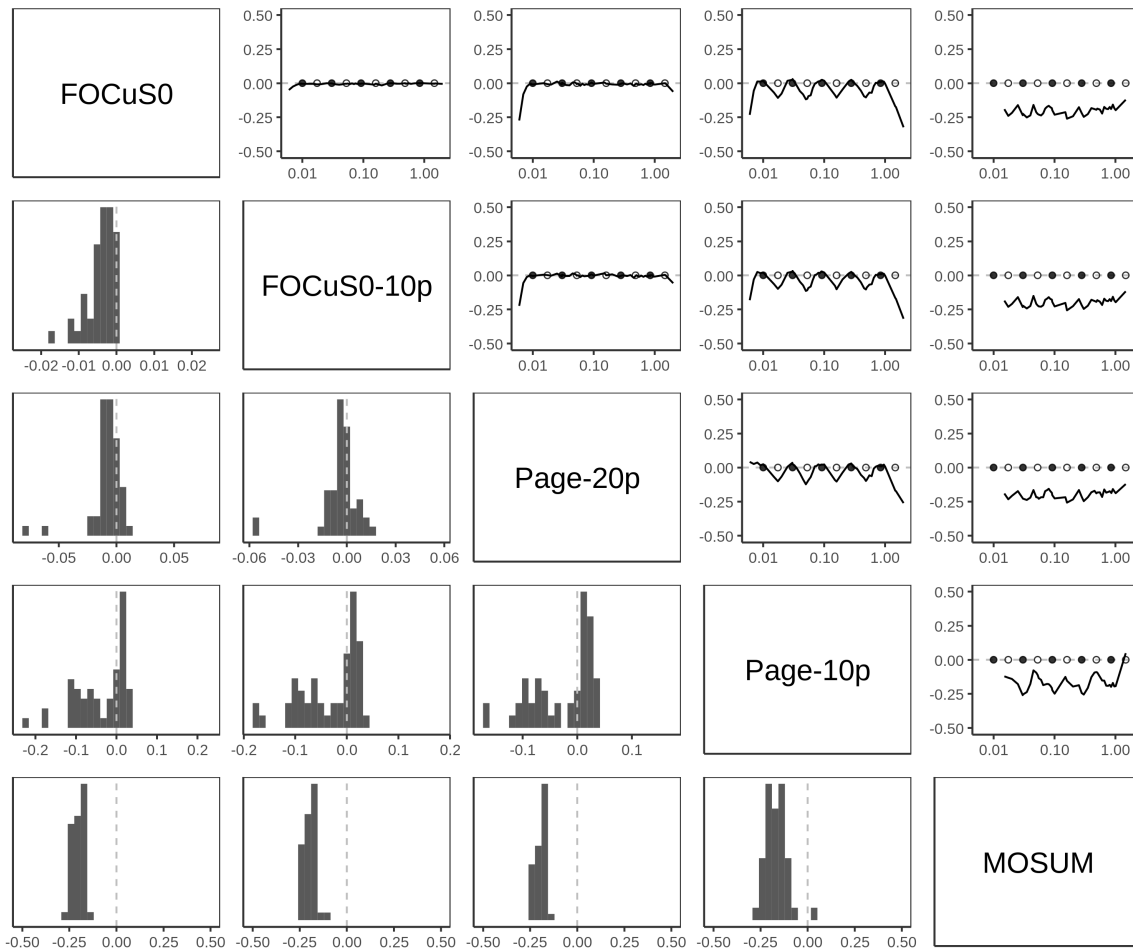
15

Figure 5: Plot matrix for log-ratios of the average detection delays of each method considered in the study. The diagonal indicates the name of the i-th tested method. The (i,j)-th plot shows the log-ratios of the average detection delays of methods i and j, with the highest index method in the numerator and the lower in the denominator, so values below zero mean the lower index method has a lower detection delay. For example, the (1, 2) and (2, 1) plots give the log-ratios between FOCuS^0 against FOCuS^0-10p, and a value below 0 would mean FOCuS^0 has the smaller average detection delay. The plots of the upper diagonal show the log-ratio as a function of the change magnitude; the dots indicate the 20 points grid, with the filled ones being the one in common with the 10 points grid. The plots on the lower diagonal show histograms of all the log-ratios, with the vertical dotted line being at 0.
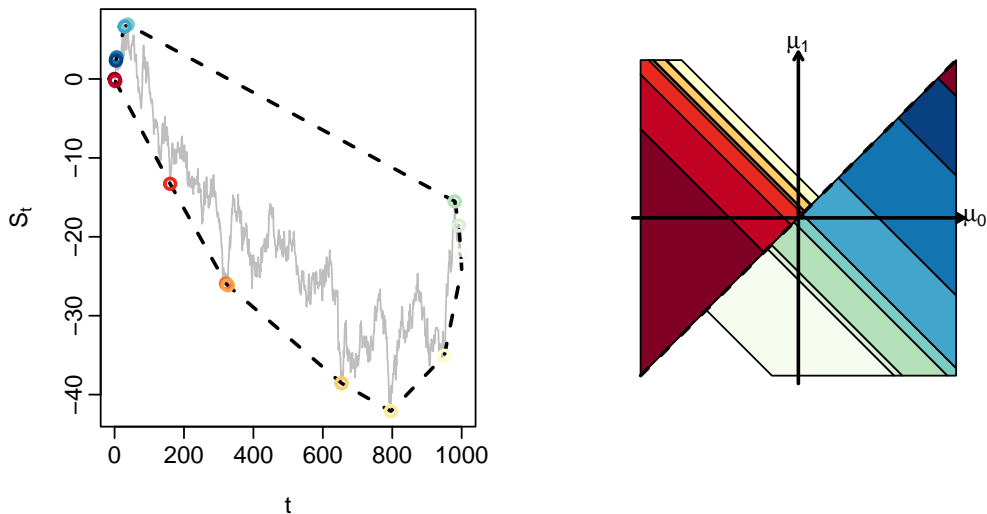
Figure 6: Convex hull of the points (t, S_t) where $S_t = \sum_{i=1}^{t} x_i$ (left-hand plot), and plot of regions in (μ_0, μ_1) space for which a different quadratic is optimal for $\mathcal{Q}_n(\mu_0, \mu_1)$ (right-hand plot). Each quadratic in the definition of $\mathcal{Q}_n(\mu_0, \mu_1)$ corresponds to a vertex on the convex hull of the random walk of the data (in that it corresponds to a change at the time associated with that vertex) and is shaded in the same colour as the corresponding vertex.

Let $\mathcal{Q}_n(\mu_0, \mu_1) = \max_{\tau \in 1,\ldots,n-1} \mathcal{Q}_{\tau,n}(\mu_0, \mu_1)$. Then the log likelihood-ratio statistic can be calculated as

$$LR_n = 2 \left\{ \max_{\mu_0, \mu_1 \in \mathbb{R}} \mathcal{Q}_n(\mu_0, \mu_1) - \max_{\mu_0 \in \mathbb{R}} \mathcal{Q}_n(\mu_0, \mu_0) \right\}.$$

Thus if we can calculate the function $\mathcal{Q}_n(\mu_0, \mu_1)$ we can calculate the log likelihood-ratio test statistic.

If we fix μ_0 and consider $\mathcal{Q}_n(\mu_0, \mu_1)$ as a function of μ_1 only, then, up to a constant, this is the function calculated in the previous section. We can calculate this separately for $\mu_1 > \mu_0$ and $\mu_1 < \mu_0$, and in each case that function will be piecewise quadratic, with the quadratics introduced at times that are on the convex hull of $S_t = \sum_{i=1}^{t} x_i$ (see Figure 3). For a quadratic introduced at time τ, the quadratic is of the form

$$-\frac{1}{2} \sum_{t=1}^{n} x_t^2 + \tau \mu_0 \left(\frac{S_\tau}{\tau} - \frac{\mu_0}{2} \right) + (n-\tau)\mu_1 \left(\frac{S_n - S_\tau}{n - \tau} - \frac{\mu_1}{2} \right).$$

Thus if we consider $\mu_1 > \mu_0$, say, then the function $\mathcal{Q}_n(\mu_0, \mu_1)$ will also be piecewise quadratic, with one quadratic corresponding to each point on the convex minorant of S_t. This is shown pictorially in Figure 6.

An algorithm to calculate the log likelihood-ratio statistic is thus similar to that of the previous section, except for two small differences.

1. For the interval update (step 1), in FOCuS^0 for up-changes we could restrict our attention to $\mu_1 \in [\mu_0, +\infty)$ (resp. $(-\infty, \mu_0]$ for down-changes), whereas in FOCuS, as

17

we do not know the value of the first segment mean, we need to consider all cases for the pre-change mean. This means that for FOCuS and up-changes we run Algorithm 2 with line 7 changed to

$$l_{k+1} \leftarrow 2(s_{k+1} - s_i)/(\tau_{k+1} - \tau_i).$$

That is we no longer take the maximum of this and the pre-change mean value. For down-changes we can apply the same algorithm but, by symmetry for data with the sign flipped, i.e. $-x_{1:n}$.

2. For the maximisation (step 2) in FOCuS^0 we only need to optimize the value of the last segment, whereas in FOCuS we also need to optimize over the pre-change mean. That is, at time n, for a quadratic defined by triple (τ_i, s_i, l_i) we calculate

$$\tau_i \left(\frac{S_i}{\tau_i} \right)^2 + (n - \tau_i) \left(\frac{S_n - S_i}{n - \tau_i} \right)^2 - n \left(\frac{S_n}{n} \right)^2.$$

We then find the maximum value of these values, maximising over all quadratics stored at time n.

By the same argument as that of Theorem 2, we have that solving the recursion has an average cost per iteration that is constant. We derive in Theorem 4 in Appendix B the same bound on the expected number of candidates as for FOCuS^0, showing that the expected per-iteration time and memory complexity of maximising the solution of FOCuS at time T is $O(\log(T))$.

3.1 Simulation Study

We make a comparison between FOCuS with the pre-change mean unknown and FOCuS^0 with the pre-change-mean known learned over a training sequence. This is because, as mentioned in the introduction, one could estimate the mean of a Gaussian process when the pre-change mean is unknown, and use such a value to run the algorithms introduced in Section 2. We study in particular the performances of FOCuS^0 as we vary the size of training data from 1000 observations up to 1×10^5. As a benchmark, we also compare to FOCuS^0 with known pre-change mean – which should show the best possible performance. We compare both average run length as a function of the threshold and detection delay as a function of the magnitude of a change. For each experiment, we report summaries over 100 replicates, and the results on detection delay are for thresholds chosen so each algorithm has an average run-length of 1×10^6. In all cases, we simulate data with 1×10^5 data points prior to the change. Results are summarised in Figure 7.

FOCuS requires a smaller threshold to achieve the same average run length of various FOCuS^0 implementations, however, differences become negligible when comparing the methods over larger training sizes. Concerning detection delay, the advantage of FOCuS is that it can improve its estimate of the pre-change mean using the data prior to any change, thus we see substantial benefits of FOCuS relative to FOCuS^0 when the amount of training data is small. However, when the amount of training data is of the same order as the amount of data prior to the change, FOCuS^0 has more power than FOCuS for detecting very small
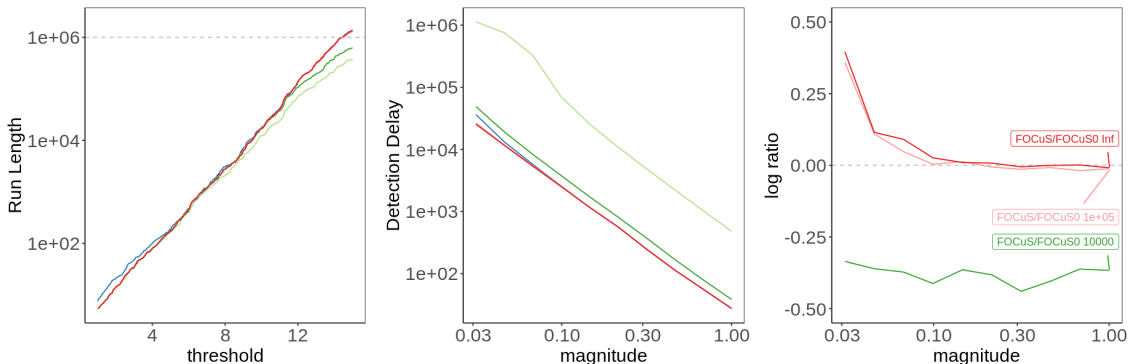
Figure 7: Comparison between FOCuS pre-change unknown and pre-change known: Average run length against threshold (left); average detection delay against magnitude of change (middle); and log-ratio of average detection delay of pairs of methods against magnitude of change (right). For the first two plots: the methods are FOCuS (blue); FOCuS^0 with pre-change mean known (red); and FOCuS^0 with different training data sizes: 1000 (light green), 1×10^4 (dark green) and 1×10^5 (pink).

changes, whereas FOCuS has more slightly more power for detecting larger changes. The fact that FOCuS is worse than FOCuS^0 with a large training dataset for small magnitudes is intuitively expected. On the one hand, the cost optimized by FOCuS is bounded, making small changes undetectable as the cost will always be smaller than the pre-defined threshold. On the other hand, the cost of FOCuS^0 is unbounded, therefore assuming we have a reasonably close estimate of the pre-change mean small changes can be identified.

4. FOCuS in the presence of outliers

Further extensions of FOCuS are to use different loss functions to the square error loss obtained from a Gaussian log-likelihood. Motivated by the application in Section 5 we will consider a robust loss function, the biweight loss, which enables us to detect changepoints in the presence of outliers (see Fearnhead and Rigaill, 2019, though other loss functions that are piecewise quadratic, such as the L_1 loss, can be used with a similar approach). The biweight loss is just the square error loss but capped at a maximum value, K, chosen by the user. To be consistent with earlier sections, we can then define an associated measure of a fit to the data, as minus this loss,

$$F(x_t, \mu_1) = -\min\left\{ \left(\frac{\mu_1}{2} - x_t\right)^2, K \right\}. \tag{12}$$

We then aim to detect a change by monitoring the resulting fit to the data if we maximise over potential locations of a change. This leads to the following functional recursion:

$$Q_n(\mu) = \max\left\{ \max_{\mu_0} \sum_{t=1}^n F(x_t, \mu_0), \ Q_{n-1}(\mu) + F(x_n, \mu) \right\}. \tag{13}$$

Using ideas described in Section 3.2 of Fearnhead and Rigaill (2019) it is straightforward to implement this recursion for all μ efficiently. For this model we are unable to recover a bound
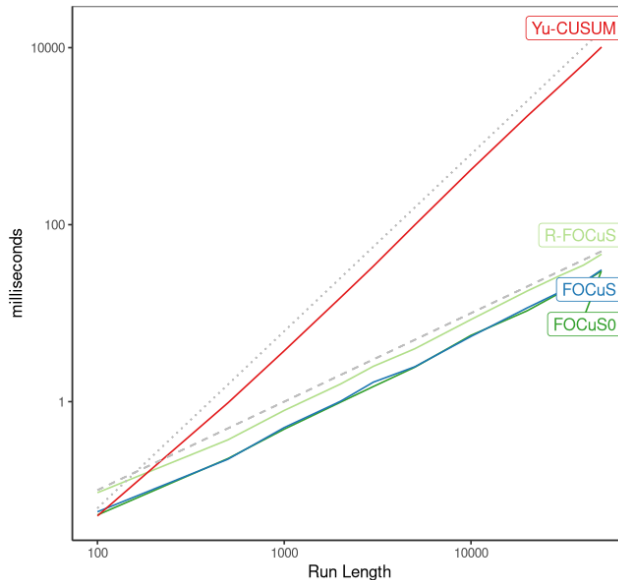
Figure 8: Runtime in milliseconds of FOCuS^0, FOCuS, R-FOCuS and Yu-CUSUM in function of the length of the sequence (log-scale on both axes). Grey lines refer to an expected $O(n)$ increase (dashed) and $O(n^2)$ increase (dotted).

on the expected number of candidate changepoints. However, we observed empirically that the cost for iteration n is in $O(\log(n))$ (see Figure 8).

In Figure 8 we give a comparison of the runtime between FOCuS^0, FOCuS, the robust implementation introduced in (13) (R-FOCuS) and Algorithm 3 from Yu et al. (2020), denoted as Yu-CUSUM. Runtimes were recorded for multiple finite sequences of lengths ranging from 100 to 5×10^4. To produce a fair comparison both implementations were written in C++, and all simulations were performed on a common personal computer. We find little difference when comparing FOCuS^0 with FOCuS, both showing an empirical linear increase in timings with the latter being slightly slower. When comparing FOCuS to Yu-CUSUM, we find a comparable runtime only up to $n = 100$, after which FOCuS is faster, due to the quadratic computational complexity of Yu-CUSUM. Lastly, we notice how R-FOCUS, while still retaining a linear computational complexity, has a larger overhead compared to simpler implementations.

5. Application of FOCuS to the AWS Cloudwatch CPU utilization

We now evaluate FOCuS by comparing with a bespoke anomaly detection algorithm on the Amazon CPU utilization datasets from the Numenta Anomaly Benchmark (Ahmad et al., 2017). The aim of these datasets is to detect anomalous behaviours in the CPU utilization of various Amazon Cloudwatch instances. For each dataset anomalous behaviours have been manually flagged by experts, and those stand as the ground truth. The data sets are shown in Figure 9, and demonstrate a range of behaviour. As point anomalies are common we will use the R-FOCuS algorithm.

Detector	Precision	Recall
R-FOCuS	0.58	0.82
Numenta HTM	0.50	0.76

Table 1: Precision and Recall for R-FOCuS and Numenta HTM.

When evaluating algorithms we will follow the methodology in Ahmad et al. (2017). A detection is deemed to be correct if it lies within $\pm 0.05 \cdot n$ of the true anomaly, where n is the length of the time series; and multiple detections within the window are allowed. A method can use the first 15% of each dataset, a portion of data known to not include any anomalies, to set tuning parameters. We use this data to tune both K in the biweight loss and the detection threshold as described in Appendix D.

As some data sets have multiple anomalies to be detected, we have to adapt R-FOCuS so that it does not stop once a change to some anomalous behaviour has occurred. To adapt R-FOCuS we simply initiate the procedure again at the estimated changepoint location after a detection is triggered. In order to reduce the number of false positives and to extend the average run length of the algorithm, at each detection we inflate the threshold by a factor of $\log(\tau_s)/\log(\tau_s - \tau_{s-1})$, with $\tau_0, ..., \tau_k$ being the estimated changepoint locations. This is the same inflation as used during the training phase, and its form is based on theory that suggests the threshold should be proportional to the log average run length (Yu et al., 2020). Inflating the penalty whilst analysing the test data is conservative, as the detected changes may be real rather than false positives, but avoids issues with bursts of false positives caused by heterogeneity in the data.

We compare R-FOCuS with numenta HTM, the best performing algorithm to date on these data. Numenta HTM (Ahmad et al., 2017) is an anomaly detection algorithm that employs an unsupervised neural network model to work with temporal data (Cui et al., 2016) to perform anomaly detection.

Results are summarised in Figure 9 and Table 1. We find that R-FOCuS has better performances in term of Precision, the proportion of true anomalies detected, and Recall, the proportion of detections that are true anomalies, compared with Numenta HTM. On a case-to-case basis, in most of the sequences both algorithms correctly flagged the anomalies. HTM overall achieves slightly shorter detection delays (with the exception of **f**), however, it produces more false positives (13 false detections against 7 of R-FOCuS). In terms of missed detections, both algorithms perform similarly, with R-FOCuS missing an anomaly flagged by HTM in **a**, and HTM missing an anomaly flagged by R-FOCuS in **d**.

Overall, this shows the flexibility of R-FOCuS for online change detection, especially considering that R-FOCuS is a simpler approach which is operating under model misspecification, and with significantly shorter computational run-times than Numenta HTM (on 3000 observations R-FOCuS takes roughly 2 milliseconds against the 4 minutes for HTM).

6. Discussion

One of the advantages of the functional pruning recursion lies in the numerous extensions possible when manipulating the cost directly. For instance, conditions on the underlying
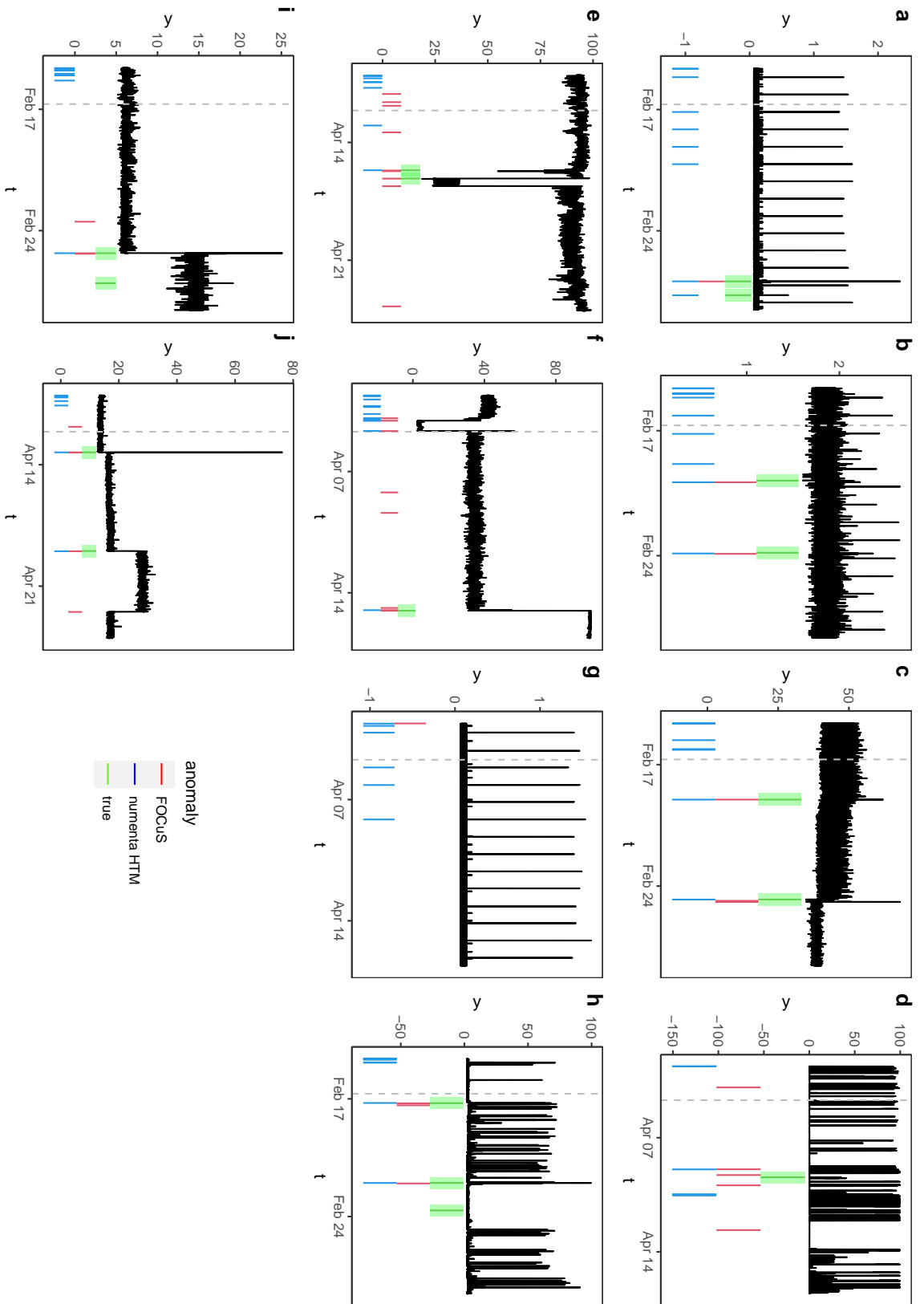
Figure 9: The 8 (**a - j**) different time series of AWS Cloudwatch CPU utilization. Green, red and blue segments correspond, respectively, to the real and estimated anomaly locations of R-FOCuS and Numenta HTM. The green rectangle around each anomaly is the anomaly window (the area in which an anomaly must be detected to count as a true positive). Multiple detections within the green lines are allowed, and are not considered as false positives. The dashed line corresponds to the probation period, used for training of tuning parameters: detections before the dashed line are not accounted for in the final result.

		Pre-change mean known		Pre-change mean unknown	
sparsity	magnitude	`ocd`	FOCuS^0	`ocd`	FOCuS
0.01	0.25	**318.30**	365.03	**1281.24**	1446.36
0.01	0.5	**95.01**	95.37	336.23	**121.29**
0.01	1	36.37	**25.15**	131.13	**26.63**
0.01	2	12.65	**7.34**	59.83	**7.38**
0.05	0.25	**503.46**	702.48	**2164.01**	2172.04
0.05	0.5	**144.57**	180.37	622.55	**317.72**
0.05	1	43.91	**43.38**	206.59	**48.66**
0.05	2	14.08	**12.40**	86.05	**12.77**
0.1	0.25	**616.11**	803.43	2348.26	**2312.23**
0.1	0.5	**171.11**	233.34	823.61	**525.72**
0.1	1	**50.88**	59.57	258.74	**69.33**
0.1	2	**14.73**	16.99	106.61	**17.64**
1	0.25	**904.74**	1110.48	2903.23	**2506.79**
1	0.5	**245.00**	420.56	1769.32	**1081.09**
1	1	**64.39**	130.30	675.58	**190.29**
1	2	**17.85**	39.74	210.92	**44.09**

Table 2: Average detection delay for changes of different sparsity levels and magnitudes in multivariate sequences. We consider two scenarios: where the pre-change mean is known and where the pre-change mean is unknown. For the former we use FOCuS^0, for the latter we use FOCuS and implement `ocd` with pre-change means estimated from training data of length 500. The sparsity specifies the proportion of the 100 data streams that change. For a change which affects k series we simulate $Z_1, \ldots, Z_k$ independent standard normal variables, and change the mean of the ith series, for $i = 1, \ldots, k$, by $mZ_i/\sqrt{\sum_{j=1}^{k} Z_j^2}$, where m is the magnitude of the change. All changes occur at time 200.

mean object of inference could be implemented to produce inference constrained to specific change patterns (Hocking et al., 2020; Runge et al., 2020), or to account for fluctuating signals and autocorrelation in the noise (Romano et al., 2020), or allow for known behaviour of the mean between changes (Jewell et al., 2020).

The main limitation of the recursions is that they rely on functional pruning ideas that currently only work for functions of a univariate parameter (though see Runge, 2020, for ideas on extending pruning to higher dimensions). However, there is still potential for applying versions of FOCuS in situations where multiple parameters may change, for example by separately testing for changes in each parameter and merging this information. A simple way of merging the FOCuS statistics is to take either their maximum or their sum (see e.g. Mei, 2010). The maximum is an appropriate test statistic for detecting a change in one or a small number of series, while the sum is appropriate for detecting a change in many series (though see Enikeeva and Harchaoui, 2019; Fisch et al., 2021; Tickle et al., 2021, for other approaches to combining statistics across data streams).

To investigate the potential for such a method we have run a simulation study comparing an approach which uses both the maximum and sum of the FOCuS statistics across data streams, and compared it to the `ocd` method of Chen et al. (2022). We used a similar simulation set-up to Chen et al. (2022), with data from 100 data streams. As suggested in Chen et al. (2022), we used simulation from a model with no change to choose thresholds for both the maximum and sum statistics, and for the three `ocd` statistics, so that both the FOCuS method and `ocd` had an average run length of 5,000. We then compared methods based on average detection delay for changes of different sizes and affecting different numbers of data streams. The results are shown in Table 2.

The `ocd` method calculates the sequential-Page statistic for a grid of change values for each stream. It then combines these using the maximum. It also uses two statistics based on averaging the log likelihood-ratio test statistic for a change at times corresponding to times of changes with large sequential-Page statistics for individual series. One of these is like our sum statistic, but it assumes the same change time in all series, unlike our implementation which sums test statistics which can correspond to changes at different times. The other statistic for combining across data streams is designed to pick up sparse changes but where multiple streams change.

The results show that if the pre-change mean is known, `ocd` tends to perform better than our simplistic way of merging the FOCuS^0 statistics. Though our approach is able to detect sparse and large changes more quickly, because the `ocd` method analyses each data stream using a grid with a relatively small largest change size. However, if the pre-change means are unknown and have to be estimated from training data of length 500, we see that accounting for this by using FOCuS leads to substantially better performance. The gain in performance will clearly depend on how much training data there is.

FOCuS is not an online algorithm, as the number of quadratics to maximise per iteration can fluctuate and is unbounded. We suggested two ways to implement an online version by maximising only at most P quadratics at each iteration based on maximising those that are optimal for a geometrically-space grid of values of the post-change mean. Such a method will be uniformly better than using sequential-Page with the same grid. However, there are other, potentially better, ways of choosing which quadratics to maximise. First one could cycle through quadratics, so that at time $t + 1$ you start maximising quadratics that were

not considered at time t. Alternatively, you could use the value of the observation to help choose which quadratics to maximise. For example, if one receives observation $x_t > 0$ then this will increase $Q_t(\mu)$ only for $\mu \in (0, 2x_t)$, and most at $\mu = x_t$. Thus there is no point to consider quadratics that are optimal only for values larger than $2x_t$, and one could give priority to quadratics that are optimal for regions of μ closest to x_t.

In line with Theorem 4, one further possible area of development would be studying the explicit distribution of the number of quadratics under the alternative, in case the statistic does not reach the threshold. In such case, we expect the number of changes to increase at a faster rate then when under the null: an additional test could be placed on the expected number of quadratics as a fail-safe mechanism to declare a change.

Following the proof of Theorem 4 it is fairly easy to show that the set of candidate changepoints stored by the offline pDPA algorithm (Rigaill, 2015) run for one change is included in the set of changepoints stored by FOCuS. This provides a bound on the expected complexity of pDPA for one change. Future work may consider extending the proof of Theorem 4 to get the expected complexity of pDPA for more than one change or for other functional pruning algorithms such as FPOP (Maidstone et al., 2017) or GFPOP (Hocking et al., 2020).

References

Joshua Simon Abramson. *Some Minorants and Majorants of Random Walks and Lévy Processes*. PhD thesis, UC Berkeley, 2012.

Subutai Ahmad, Alexander Lavin, Scott Purdy, and Zuha Agha. Unsupervised real-time anomaly detection for streaming data. *Neurocomputing*, 262:134–147, 2017.

Joham Alvarez-Montoya, Alejandro Carvajal-Castrillón, and Julián Sierra-Pérez. In-flight and wireless damage detection in a UAV composite wing using fiber optic sensors and strain field pattern recognition. *Mechanical Systems and Signal Processing*, 136:106526, 2020.

Erik Sparre Andersen. On the fluctuations of sums of random variables ii. *Mathematica Scandinavica*, 2:195–223, 1955.

Michele Basseville and Igor V Nikiforov. *Detection of abrupt changes: theory and application*, volume 104. Prentice Hall Englewood Cliffs, 1993.

Michele Basseville, Albert Benveniste, Maurice Goursat, and Laurent Meve. In-flight vibration monitoring of aeronautical structures. *IEEE Control Systems Magazine*, 27(5): 27–42, 2007.

Yudong Chen, Tengyao Wang, and Richard J Samworth. High-dimensional, multiscale online changepoint detection. *Journal of the Royal Statistical Society (Series B)*, 84: 234–266, 2022.

Zhanshou Chen and Zheng Tian. Modified procedures for change point monitoring in linear models. *Mathematics and Computers in Simulation*, 81(1):62–75, 2010.

Chia-Shang J Chu, Kurt Hornik, and Chung-Ming Kaun. MOSUM tests for parameter constancy. *Biometrika*, 82(3):603–617, 1995.

Gari D Clifford, Ikaro Silva, Benjamin Moody, Qiao Li, Danesh Kella, Abdullah Shahin, Tristan Kooistra, Diane Perry, and Roger G Mark. The PhysioNet/computing in cardiology challenge 2015: reducing false arrhythmia alarms in the ICU. In *2015 Computing in Cardiology Conference (CinC)*, pages 273–276. IEEE, 2015.

Yuwei Cui, Subutai Ahmad, and Jeff Hawkins. Continuous online sequence learning with an unsupervised neural network model. *Neural Computation*, 28(11):2474–2504, 2016.

Peter Eiauer and Peter Hackl. The use of MOSUMS for quality control. *Technometrics*, 20 (4):431–436, 1978.

Farida Enikeeva and Zaid Harchaoui. High-dimensional change-point detection under sparse alternatives. *The Annals of Statistics*, 47(4):2051–2079, 2019.

Paul Fearnhead and Guillem Rigaill. Changepoint detection in the presence of outliers. *Journal of the American Statistical Association*, 114(525):169–183, 2019. ISSN 0162-1459.

Alexander TM Fisch, Idris A Eckley, and Paul Fearnhead. Subset multivariate collective and point anomaly detection. *Journal of Computational and Graphical Statistics*, pages 1–12, 2021.

PA Fridman. A method of detecting radio transients. *Monthly Notices of the Royal Astronomical Society*, 409(2):808–820, 2010.

Fabio Fuschino, Riccardo Campana, Claudio Labanti, Yuri Evangelista, Marco Feroci, L Burderi, Fabrizio Fiore, Filippo Ambrosino, G Baldazzi, and P Bellutti. HERMES: An ultra-wide band X and gamma-ray transient monitor on board a nano-satellite constellation. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 936:199–203, 2019.

Josua Gösmann, Tobias Kley, and Holger Dette. A new approach for open-end sequential change point monitoring. *arXiv preprint arXiv:1906.03225*, 2019.

Toby Hocking, Guillem Rigaill, Paul Fearnhead, and Guillaume Bourque. Constrained dynamic programming and supervised penalty learning algorithms for peak detection in genomic data. *Journal of Machine Learning Research*, 21:1–40, 2020.

Daniel R Jeske, Nathaniel T Stevens, Alexander G Tartakovsky, and James D Wilson. Statistical methods for network surveillance. *Applied Stochastic Models in Business and Industry*, 34(4):425–445, 2018.

Sean W Jewell, Toby Dylan Hocking, Paul Fearnhead, and Daniela M Witten. Fast non-convex deconvolution of calcium imaging data. *Biostatistics*, 21(4):709–726, 2020. doi: 10.1093/biostatistics/kxy083.

Claudia Kirch and Joseph Tadjuidje Kamgaing. On the use of estimating functions in monitoring time series for change points. *Journal of Statistical Planning and Inference*, 161:25–49, 2015.

Claudia Kirch and Silke Weber. Modified sequential change point procedures based on estimating functions. *Electronic Journal of Statistics*, 12(1):1579–1613, 2018.

G. Lorden. Procedures for reacting to a change in distribution. *Ann. Math. Statist.*, 42(6): 1897–1908, 1971.

Robert Maidstone, Toby Hocking, Guillem Rigaill, and Paul Fearnhead. On optimal multiple changepoint algorithms for large data. *Statistics and Computing*, 27(2):519–533, Mar 2017.

Yajun Mei. Efficient scalable schemes for monitoring a large number of data streams. *Biometrika*, 97(2):419–433, 2010.

Avraham A Melkman. On-line construction of the convex hull of a simple polyline. *Information Processing Letters*, 25(1):11–12, 1987.

Ewan S Page. Continuous inspection schemes. *Biometrika*, 41(1/2):100–115, 1954.

Ewan S Page. A test for a change in a parameter occurring at an unknown point. *Biometrika*, 42(3/4):523–527, 1955.

Tao Peng, Christopher Leckie, and Kotagiri Ramamohanarao. Proactively detecting distributed denial of service attacks using source IP address monitoring. In *International Conference on Research in Networking*, pages 771–782. Springer, 2004.

Theodor D Popescu and Dorel Aiordăchioaie. New procedure for change detection operating on Rényi entropy with application in seismic signals processing. *Circuits, Systems, and Signal Processing*, 36(9):3778–3798, 2017.

AD Pouliezos and George S Stavrakakis. *Real time fault monitoring of industrial processes*, volume 12. Springer Science & Business Media, 2013.

Marion R Reynolds. Approximations to the average run length in cumulative sum control charts. *Technometrics*, 17(1):65–71, 1975.

Guillem Rigaill. A pruned dynamic programming algorithm to recover the best segmentations with 1 to kmax change-points. *Journal de la Societe Francaise de Statistique*, 156 (4):180–205, 2015.

Gaetano Romano, Guillem Rigaill, Vincent Runge, and Paul Fearnhead. Detecting abrupt changes in the presence of local fluctuations and autocorrelated noise. *Journal of the American Statistical Association*, 2020.

Vincent Runge. Is a finite intersection of balls covered by a finite union of balls in Euclidean spaces? *Journal of Optimization Theory and Applications*, 187(2):431–447, 2020.

Vincent Runge, Toby Dylan Hocking, Gaetano Romano, Fatemeh Afghah, Paul Fearnhead, and Guillem Rigaill. gfpop: an R package for univariate graph-constrained change-point detection. *arXiv preprint arXiv:2002.03646*, 2020.

Alexander G Tartakovsky, Aleksey S Polunchenko, and Grigory Sokolov. Efficient computer network anomaly detection by changepoint detection methods. *IEEE Journal of Selected Topics in Signal Processing*, 7(1):4–11, 2012.

Sam O Tickle, IA Eckley, and Paul Fearnhead. A computationally efficient, high-dimensional multiple changepoint procedure with application to global terrorism incidence. *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, 2021.

Venugopal V Veeravalli and Taposh Banerjee. Quickest change detection. In *Academic Press Library in Signal Processing*, volume 3, pages 209–255. Elsevier, 2014.

Liyan Xie, Yao Xie, and George V Moustakides. Asynchronous multi-sensor change-point detection for seismic tremors. In *2019 IEEE International Symposium on Information Theory (ISIT)*, pages 787–791. IEEE, 2019.

Yi Yu, Oscar Hernan Madrid Padilla, Daren Wang, and Alessandro Rinaldo. A note on online change point detection. *arXiv preprint arXiv:2006.03283*, 2020.

SUPPLEMENTARY MATERIAL

Appendix A. Proof of Proposition 1

It is straightforward to show, for example by induction, that the solution $Q_n(\mu)$ to recursion (6) can be written in the form

$$Q_n(\mu) = \max_{s=0,\ldots,n} \left\{ \sum_{t=s+1}^{n} \mu \left(x_t - \frac{\mu}{2} \right) \right\},$$

where we treat the sum from $n+1$ to n as empty, and hence equal to 0. Hence

$$\max_{\mu} Q_n(\mu) = \max_{s=0,\ldots,n} \left\{ \max_{\mu} \sum_{t=s+1}^{n} \mu \left(x_t - \frac{\mu}{2} \right) \right\}$$

$$= \max_{s=0,\ldots,n-1} \left\{ \frac{1}{2} \sum_{t=s+1}^{n} \left(\frac{\sum_{j=s+1}^{n} x_j}{n-s} \right)^2 \right\}$$

$$= \max_{w=1,\ldots,n} \left\{ \frac{1}{2} w \left(\frac{\sum_{j=n-w+1}^{n} x_j}{w} \right)^2 \right\}$$

The second line uses the fact that the maximum over μ is when μ is the sample mean of $x_{\tau:n}$. For the second step we use the fact that the maximum is never the empty sum, and thus we can drop the case $s = n$ from the maximisation. The terms in the final expression are just $(1/2) M_w(n)^2$ as required. The result in terms of $P(n)$ follows directly from $P(n) = \max_w |M_w(n)|$. $\square$

Appendix B. On the expected number of changes stored by FOCuS

B.1 Variants of the FOCuS implementations

We study the number of candidate changepoints $\tau \in \{1, \cdots, n\}$ stored by FOCuS at each iteration. We report the possible FOCuS optimizations introduced in the main body of the paper:

- FOCuS^0 which solves the problem for a known pre-change mean μ_0 (typically 0) and unknown post-change mean μ_1.

$$\mathcal{Q}_n^0 = \max_{\substack{\tau \in \{1,\ldots,n\} \\ \mu_0=0, \mu_1 \in \mathbb{R}}} \left\{ -\sum_{t=1}^{\tau} (x_t - \mu_0)^2 - \sum_{t=\tau+1}^{n} (x_t - \mu_1)^2 \right\}. \tag{14}$$

  This problem is solved through Algorithm 2, an algorithm similar to Melkman's algorithm (Melkman, 1987).

- FOCuS which solves the problem for both unknown pre-change and post-change means:

$$\mathcal{Q}_n = \max_{\substack{\tau \in \{1,\ldots,n\} \\ \mu_1, \mu_0 \in \mathbb{R}}} \left\{ -\sum_{t=1}^{\tau} (x_t - \mu_0)^2 - \sum_{t=\tau+1}^{n} (x_t - \mu_1)^2 \right\}. \tag{15}$$

B.2 Assumptions and definitions

In the rest of this section, we let $x_1, \ldots x_n$ be our ordered sequence of observations. To denote a subset of such a sequence, we will write $x_{i:j} = x_i, \ldots, x_j$ for $i < j$. We denote a true changepoint with τ^*.

We assume that:

$$x_i = \mu_i + \varepsilon_i, \tag{16}$$

where ε_i are i.i.d with a continuous distribution and μ_i is a piecewise constant signal in 1 or 2 pieces.

We define the cost of a segmentation $x_{i:j}$ with a change at τ as:

$$q_{i:j,\tau}(\mu_0, \mu_1) = \sum_{t=i}^{\tau} (x_t - \mu_0)^2 + \sum_{t=\tau+1}^{j} (x_t - \mu_1)^2.$$

with pre-change and post-change means μ_0 and μ_1. As a convention, for $j = \tau$, we take: $q_{i:j,j}(\mu_0, \mu_1) = \sum_{t=i}^{j}(x_t - \mu_0)^2$.

Sets of candidate changepoints   We call $\mathcal{I}_{i:j}^0$ the set of candidate changepoints stored by FOCuS^0 for $\mu_1 > 0$, and $\mathcal{I}_{i:j}$ the set stored by FOCuS for $\mu_1 > \mu_0$. By definition those will be:

$$
\begin{aligned}
\mathcal{I}_{i:j}^0 &= \left\{ \tau \mid \exists\, \mu_1 > 0,\ \forall \tau' \neq \tau, \quad q_{i:j,\tau}(0, \mu_1) < \quad q_{i:j,\tau'}(0, \mu_1) \right\}, \\
\mathcal{I}_{i:j} &= \left\{ \tau \mid \exists\, \mu_1 > \mu_0, \forall \tau' \neq \tau, \quad q_{i:j,\tau}(\mu_0, \mu_1) < \quad q_{i:j,\tau'}(\mu_0, \mu_1) \right\},
\end{aligned}
$$

By definition $\mathcal{I}_{i:j}^0$ is in $\mathcal{I}_{i:j}$ Our goal will then be to control the size of the set $\mathcal{I}_{i:j}$ of candidate changepoints stored by FOCuS.

B.3 Main results

On the assumption of a realization from (16) we can get the following bound on the number of changepoints stored by FOCuS^0 and FOCuS.

Theorem 4 *For all $n \geq 1$*

$$E(\#\mathcal{I}_{1:n}^0) \leq E(\#\mathcal{I}_{1:n}).$$

If μ_i is constant with respect to i:

$$E(\#\mathcal{I}_{1:n}) = 1 + \sum_{1}^{n-1} 1/(t+1) \leq (1 + \log(n))$$

and if μ_i has a single changepoint, we have

$$E(\#\mathcal{I}_{1:n}) \leq 2(1 + \log(n/2)).$$

By symmetry, the same result holds for the number of quadratics stored by FOCuS^0 for $\mu_1 < 0$ and FOCuS for $\mu_1 < \mu_0$.

2

Overview of the proof The proof to this theorem relies on a combination of three lemmas, summarized here:

1. Lemma 5: for $i \leq j < k$ we have $\mathcal{I}_{i:k} \subseteq \mathcal{I}_{i:j} \cup \mathcal{I}_{j+1:k}$;

2. Lemma 6: $\mathcal{I}_{i:j}$ are the extreme points of the largest convex minorant of the sequence $S_{i:j}$, where $\quad S_t = \sum_{k=1}^{t} x_k$;

3. Lemma 7: that controls the number of extreme point of a random-walk (derived from Andersen, 1955; Abramson, 2012).

The three lemmas are covered in detail and proven in Appendix B.4.

Proof: By definition $\mathcal{I}_{i:j}$ includes $\mathcal{I}_{i:j}^0$ and we get the first inequality. For the case where μ_i is constant with respect to i we apply Lemma 7. For the case where μ_i has a single changepoint, using Lemma 5 we get that

$$\mathcal{I}_{1:n} \subseteq \mathcal{I}_{1:\tau^*} \cup \mathcal{I}_{\tau^*+1:n}.$$

We then apply Lemma 7 on $\mathcal{I}_{1:\tau^*}$ and $\mathcal{I}_{\tau^*+1:n}$. The worst case is obtained for $\tau^* = n/2$. $\square$
An empirical evaluation of this bound can be found in Appendix C.2.

B.4 Inclusion and convex hull Lemmas

A useful identity For any $i \leq \tau < \tau' \leq j$, μ_0 and μ_1 we have that

$$q_{i:j,\tau}(\mu_0, \mu_1) - q_{i:j,\tau'}(\mu_0, \mu_1) = (\mu_0 - \mu_1)\left(2\sum_{\tau+1}^{\tau'} x_t - (\tau' - \tau)(\mu_0 + \mu_1)\right), \qquad (17)$$

which does not depend on i and j. This identity simplifies the proof of the following lemmas.

Lemma 5 *For $i \leq j \leq k$*

$$\mathcal{I}_{i:k} \quad \subseteq \quad \mathcal{I}_{i:j} \cup \mathcal{I}_{j+1:k} \qquad (18)$$

Proof: Consider any τ in $(i+1:j) \cap \mathcal{I}_{i:k}$, by definition

$$\exists \, \mu_0 < \mu_1, \forall \, \tau' \neq \tau \; with \; \tau' \; in \; (i+1:k), \quad q_{i:k,\tau}(\mu_0, \mu_1) < q_{i:k,\tau'}(\mu_0, \mu_1),$$

then using equation (17) we get $q_{i:j,\tau}(\mu_0, \mu_1) < q_{i:j,\tau'}(\mu_0, \mu_1)$ and thus τ is also in $\mathcal{I}_{i:j}$. We proceed similarly for any τ in $(j+1:k) \cap \mathcal{I}_{i:k}$. $\square$
The next lemma relates the set $\mathcal{I}_{i:j}$ to the lower convex hull of $S_{i:j}$.

Lemma 6 *The set of τ in $\mathcal{I}_{i:j}$ are the extreme points of the largest convex minorant of the sequence $S_{i:j}$.*

Proof: Assume that τ' is in $\mathcal{I}_{i:j}$. Then, by definition of $\mathcal{I}_{i:j}$, there exists a (μ_0, μ_1) with $\mu_1 > \mu_0$ such that for all $\tau < \tau'$,

$$q_{i:j,\tau}(\mu_0, \mu_1) - q_{i:j,\tau'}(\mu_0, \mu_1) > 0$$

3

and for all $\tau'' > \tau'$

$$q_{i:j,\tau'}(\mu_0, \mu_1) - q_{i:j,\tau''}(\mu_0, \mu_1) < 0.$$

Using equation (17), the equation for $\tau < \tau'$ gives

$$(\mu_0 - \mu_1) \left(2 \sum_{\tau+1}^{\tau'} x_t - (\tau' - \tau)(\mu_0 + \mu_1) \right) \;>\; 0$$

$$\Rightarrow \left(2 \sum_{\tau+1}^{\tau'} x_t - (\tau' - \tau)(\mu_0 + \mu_1) \right) \;<\; 0$$

$$\Rightarrow \bar{x}_{\tau+1:\tau'} < \frac{1}{2}(\mu_0 + \mu_1),$$

where $\bar{x}_{\tau+1:\tau'} = \frac{\sum_{t=\tau+1}^{\tau'} x_t}{\tau'-\tau}$. The second inequality follows from $\mu_0 - \mu_1 < 0$.

A similar argument for $\tau'' > \tau'$ gives that

$$\bar{x}_{\tau'+1:\tau''} > \frac{1}{2}(\mu_0 + \mu_1).$$

For such a (μ_0, μ_1) to exist such that these inequalities hold for all $\tau < \tau' < \tau''$ we need that

$$\bar{x}_{\tau+1:\tau'} < \bar{x}_{\tau'+1:\tau''},$$

for all $\tau < \tau' < \tau''$.

Re-writing the sample means in terms of the cumulative sums of the data, this is equivalent to, for all τ and τ'' satisfying $\tau < \tau' < \tau''$:

$$\frac{S_{\tau'} - S_\tau}{\tau' - \tau} < \frac{S_{\tau''} - S_{\tau'}}{\tau'' - \tau'},$$

and therefore $(S_{\tau'}, \tau')$ is part of the largest convex minorant of the sequence $S_{i:j}$.

We now prove the converse. Assume that τ' is part of the convex minorant of $S_{i:j}$. Then for all τ and τ'' satisfying $\tau < \tau' < \tau''$:

$$\frac{S_{\tau'} - S_\tau}{\tau' - \tau} < \frac{S_{\tau''} - S_{\tau'}}{\tau'' - \tau'},$$

and therefore there exists a m such that

$$\max_\tau \bar{x}_{\tau+1:\tau'} < m < \min_{\tau''} \bar{x}_{\tau'+1:\tau''}.$$

Now picking μ_0 and μ_1 such that $\frac{1}{2}(\mu_0 + \mu_1) = m$ and $\mu_0 - \mu_1 = -1$ we get using equation (17) we get that for all $\tau < \tau'$,

$$q_{i:j,\tau}(\mu_0, \mu_1) - q_{i:j,\tau'}(\mu_0, \mu_1) > 0$$

and for all $\tau'' > \tau'$

$$q_{i:j,\tau'}(\mu_0, \mu_1) - q_{i:j,\tau''}(\mu_0, \mu_1) < 0.$$
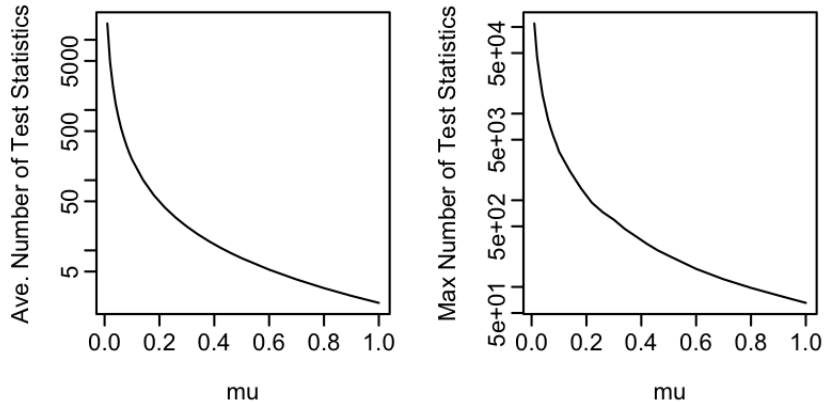
$\square$

The next lemma is based on Andersen (1955).

4

Figure 10: Plot of expected and maximum number of test statistics stored/evaluated by Lorden (1971) for data of length 10^6, against the minimum change size (mu). The y-axes are on a log-scale.

Lemma 7 *Assuming the x_t follow an i.i.d continuous distribution then $E(\#\mathcal{I}_{i:j}) = \sum_{t=1}^{j-i-1} 1/(t+1) + 1 \leq \log(n) + 1$.*

Proof: We use Lemma 6 and then apply Andersen (1955) (definitions at pages 195 and 196, then result of page 217). Andersen (1955) does not include the vertex at the final point of the random walk, so we need to add a 1 for the point at j. Standard results for the harmonic series show that $\sum_{t=1}^{j-i-1} 1/(t+1) \leq \log(j-i)$. The proof follows as $(j-i) \leq n$. □

Appendix C. Additional Empirical Results

C.1 Comparison to Lorden (1971)

We implemented Lorden (1971)'s algorithm for detecting a positive change. This algorithm requires the specification of a minimum change size, μ^* say. It then runs the sequential-Page procedure for μ^*. This procedure will reset, i.e. the sequential-Page statistic will be 0, at certain times. At these times, the sequential-Page statistic calculated for any change $\mu > \mu^*$ would be 0. Thus to detect changes bigger than μ^* Lorden (1971) proposes calculating the likelihood ratio test statistic for a change starting at any time since the last reset. I.e., if at time t the last reset was at time τ, then it would calculate the likelihood ratio statistic for a change at each of times $\tau, \tau + 1, \ldots, t - 1$. The final test statistic is the maximum over these. This involves calculating $t - \tau$ statistics, and is equivalent in cost to maximising $t - \tau$ quadratics in FOCuS^0.

Figure 10 shows the average and maximum number of test statistics stored by Lorden (1971)'s algorithm as a function of the minimum jump size. These have been calculated empirically for data of length $n = 10^6$, with the results averaged across 50 replications. We also show the expected number of statistics stored as a function of t for different minimum change sizes, and the expected number of quadratics stored by FOCuS^0 in Figure 11
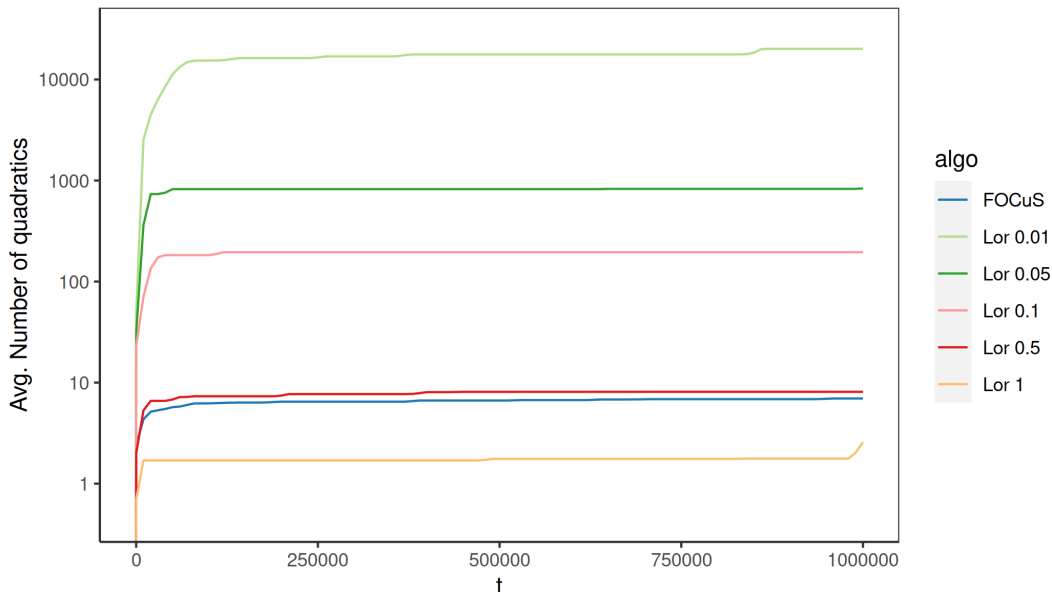
Figure 11: Plot of the expected number of test statistics stored/evaluated by Lorden (1971) and expected number of quadratics stored by $FOCuS^0$ against time for data of length 10^6. Results were obtained by fitting a monotone mean function to data from 50 replication of each algorithm. The y-axis is on a log-scale.

For data of length 10^6, $FOCuS^0$ would store around 8 quadratics on average. Thus to be computationally more efficient, Lorden (1971) would require a minimum change size of around 0.5 or higher. Furthermore, the variability in the number of test statistics stored by Lorden (1971) is much higher than for FOCuS (e.g. for a minimum change size of 0.8 it can require evaluating over 100 test statistics) which can be problematic if there is a limit on the computational resource available per iteration.

C.2 Empirical bound evaluation

To illustrate the bound of Theorem 4 we simulate signals of various lengths n (from $n = 2^{10}$ to $n = 2^{22}$) without change (100 replicates) and with one change (100 replicates). We then record the number of candidates stored by FOCuS (left) and $FOCuS^0$ (right). Where a change was present, its location was sampled uniformly between 1 and $n-1$. Similarly, the change magnitude was sampled uniformly at random in $[0, 4]$. Results are summarised in Figure 12.

The bound in Theorem 4 applies to both $FOCuS^0$ and FOCuS for an up change, and a similar bound holds by symmetry for down changes. These give a total bound on the number of quadratics stored that is $2(\log(n) + 1)$ if the data is simulated without a change and $4(\log(n/2) + 1)$ if the data is simulated with a single change. We plot the observed number of quadratics stored against $4(\log(n) + 1)$, and also plot the lines $2(\log(n) + 1)$ and $\log(n) + 1$. For FOCuS we see that the bound of $2(\log(n) + 1)$ for the case where the data is simulated without a change is tight. However, the number of quadratics stored is very

similar for data simulated with a change – suggesting that the $4(\log(n/2) + 1)$ bound for that case is conservative. For FOCuS^0 the number of quadratics stored is empirically half that stored by FOCuS for the case where the data is simulated without a change, in line with the discussion after Theorem 4. The bound in the theorem is based on a bound on the number of vertices on the convex minorant of the random walk defined by the cumulative sum of the data. However, for FOCuS^0 the number of quadratics stored is equal to a subset of the vertices, namely the vertices that are followed by a side of the minorant with positive slope. By symmetry, this would be the vertex at the minimum of the convex minorant plus, in expectation, half the remaining vertices.
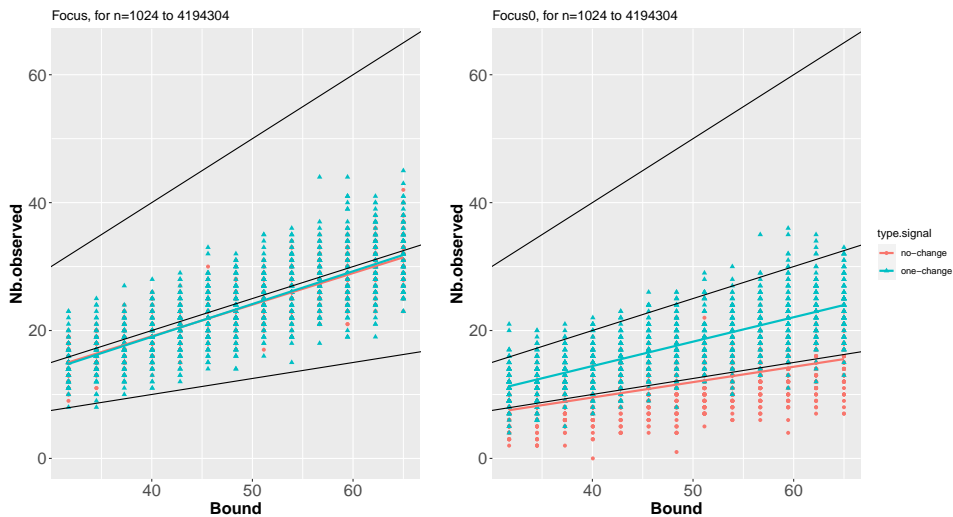

Figure 12: Number of observed quadratics stored by FOCuS (left) and FOCuS^0 (right) for signals with no change or with one change against the bound $4(\log(n) + 1)$. The three black lines represent the function $y = x$, $y = 0.5x$ and $y = 0.25x$, representing the lines $4(\log(n) + 1)$, $2(\log(n) + 1)$ and $\log(n) + 1$ respectively. Red and blue lines are the fitted regression lines for respectively the no-change and single-change scenarios.

Appendix D. Estimation of initial parameters

We propose a simple sequential for the initial parameters needed to run R-FOCuS (and the other implementations) in a semi-supervised manner. The general idea is to first find the value of the K parameter of the bi-weight loss and based on that tune the λ threshold value on a probation period. Say for each sequence we consider the first w observations for training. For tuning the K parameter of the bi-weight loss, one could simply store the initial w values and, if any observation within 1.5 interquartile ranges of those values are present, i.e. if we are in presence of outliers, then simply pick the K highest occurring value within this limit. Over the same period, we estimate the variance σ^2 and run the R-FOCuS procedure on the normalised values of the probation period with this value of K, recording the trace of the statistics at each iteration $\mathcal{Q}_1, \ldots, \mathcal{Q}_w$. We then set $\lambda = \kappa \times \max_{i=1,\ldots,w} \mathcal{Q}_i$ for some $\kappa \in \mathbb{R}^+$. In the presented application we set $\kappa = 1.5$ at the beginning and

$\kappa = \log(\tau_s)/\log(\tau_s - \tau_{s-1})$ when restarting the algorithm after a detection, with τ_s, τ_{s-1} being the last and previous last stopping times.

In total, this estimation adds a linear in w computational overhead, which consists of temporarily storing the initial w values and performing the estimation of the K.